Lecture 3: Data Modeling

Falco J. Bargagli Stoffi

10/06/2020

Lecture 3: Data Modeling

The goal of a model is to provide a simple, low-dimensional, interpretable summary of a dataset. Models are a really useful way to help you peel back layers of structure as you are exploring your dataset. Every statistical model can be "divided" in two parts: 1. a family of models that express a prece, but generic, pattern that you want to capture (i.e., the pattern can be a straight line or a quadratic curve); 2. a fitted model, that can be found by selecting the family of models that is the closest to your data.

It is important to understand that a fitted model is just the closest model from a family of models. This implies that you have the "best" model according to some criteria and based on a set of assumptions. This does not imply that your model is a good model or that your model is "true". George Box, a famous british statistician, once said one of the most quoted statistical quotes:"all models are wrong, but some are useful". It is worth reading the fuller context of the quote as it is quite illustrative of the philosophy behind any statistical model: "Now it would be very remarkable if any system existing in the real world could be exactly represented by any simple model. However, cunningly chosen parsimonious models often do provide remarkably useful approximations. For example, the law PV = RT relating pressure P, volume V and temperature T of an"ideal" gas via a constant R is not exactly true for any real gas, but it frequently provides a useful approximation and furthermore its structure is informative since it springs from a physical view of the behavior of gas molecules. For such a model there is no need to ask the question "Is the model true?" If "truth" is to be the "whole truth" the answer must be "No." The only question of interest is "Is the model illuminating and use- ful?"

This does not mean that all the models are wrong and, we should just go for the least wrong model. This quote should be interpeted as a call for careful laying down the assumptions on which the quality of the model is built on. As Berkeley statisticain Mark Van Der Laan stated in a recent article "The statistical formulation and theory should define the algorithm" source.

In this lecture we will go see how to perform in R two types of models: 1. linear regression models; 2. non-linear predictive models (decision trees, random forests).

As you have probably seen the former in your econometric classes, I will skip the mathematical details, and focus on the the function used to build these models and on the intepretation of their outputs.

```r
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------- tidyverse 1.2.1 --

## <U+2713> ggplot2 3.2.0     <U+2713> purrr   0.3.2
## <U+2713> tibble  2.1.3     <U+2713> dplyr   0.8.5
## <U+2713> tidyr   1.0.2     <U+2713> stringr 1.4.0
## <U+2713> readr   1.3.1     <U+2713> forcats 0.4.0

## -- Conflicts ------------------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(modelr)
library(hdm)
library(stabs)
```

```
## Loading required package: parallel
```

```r
library(AER)
```

```
## Loading required package: car

## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode

## The following object is masked from 'package:purrr':
##
##     some

## Loading required package: lmtest

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich

## Loading required package: survival
```

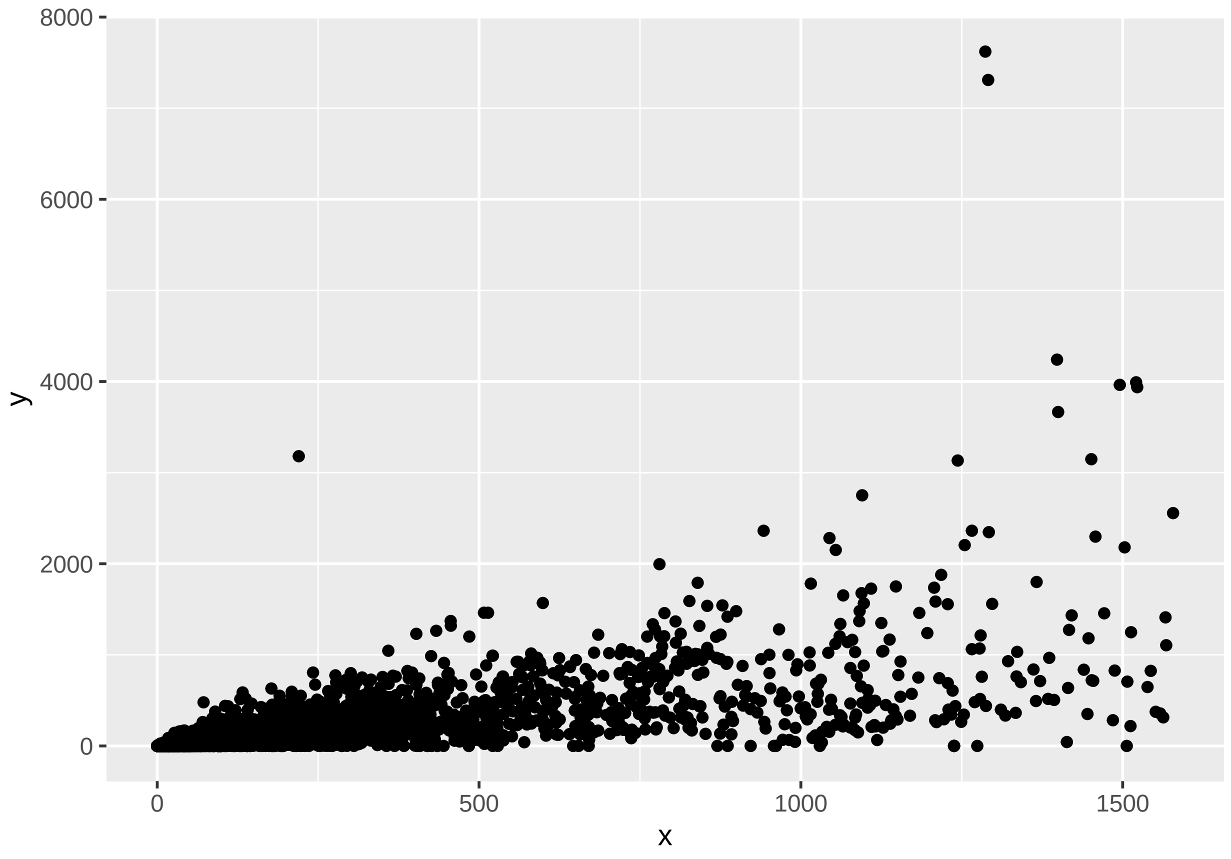
```r
library(sandwich)
library(lmtest)
library(broom)
```

```
##
## Attaching package: 'broom'

## The following object is masked from 'package:modelr':
##
##     bootstrap
```

```r
library(readxl)
data <- read_excel("G:\\Il mio Drive\\Econometrics Lab\\Data\\Compustat Data.xlsx")
data <- data[, !names(data) %in% c("Interest Expense - Total (Financial Services)",
                                   "Net Interest Income", "Nonperforming Assets - Total")]
data_clean <- na.omit(data)
```

```r
x <- data_clean$`Assets - Total`[which(data_clean$`Assets - Total`<
                                 quantile(data_clean$`Assets - Total`, 0.95))]
y <- data_clean$`Sales/Turnover (Net)`[which(data_clean$`Assets - Total`<
                                 quantile(data_clean$`Assets - Total`, 0.95))]

reg_data <- as.data.frame(cbind(x, y))
```

```
ggplot(reg_data, aes(x, y)) +
  geom_point()
```

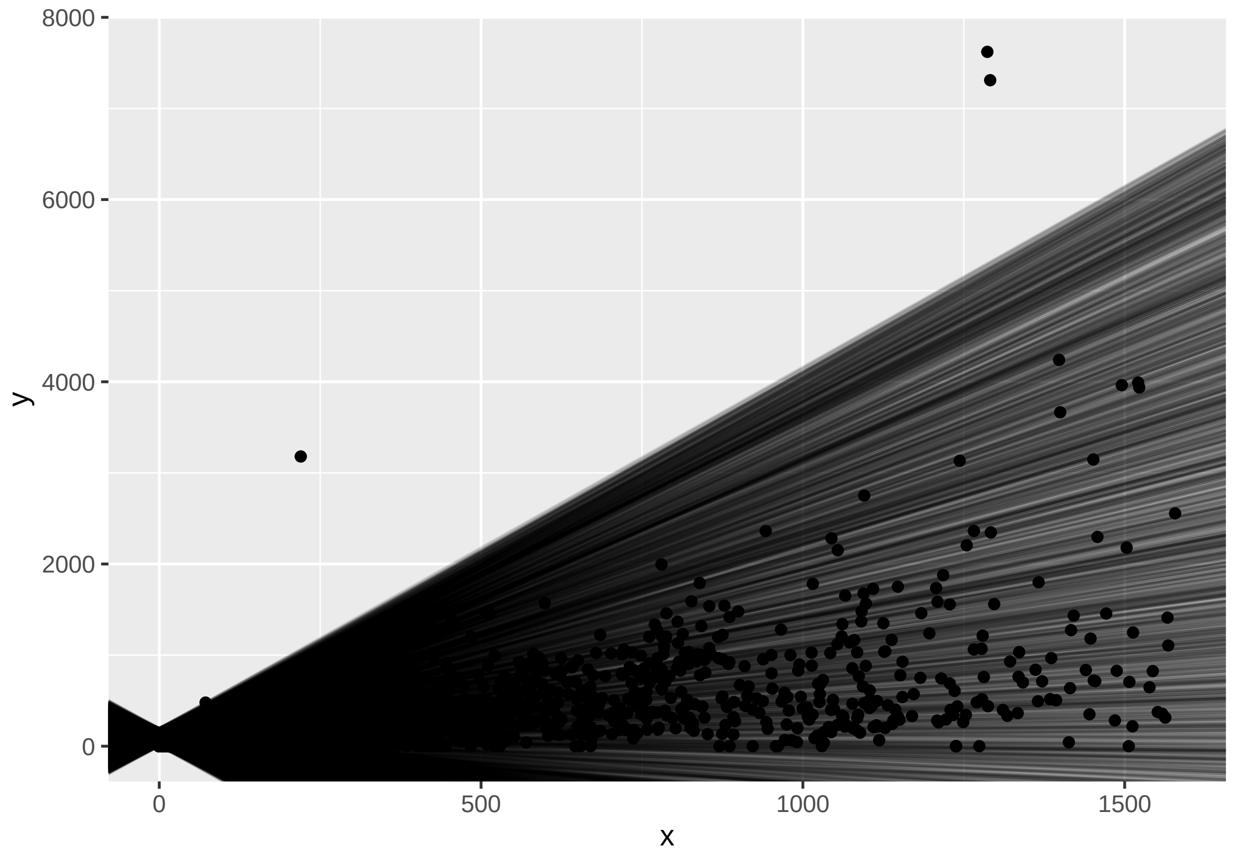


You can see a quite clear pattern in the data. Let's now use a model to capture the pattern and make it more explicit.

Let's first generate a set of random model an let's overlay them on the data.

```
models <- tibble(
  beta1 = runif(length(x), 0, 200),
  beta2 = runif(length(x), -4, 4)
)
```

```
ggplot(reg_data, aes(x, y)) +
  geom_abline(
    aes(intercept = beta1,
        slope = beta2),
    data = models, alpha = 1/15
  ) +
  geom_point()
```



```
model1 <- function(beta, data){
  beta[1] + data$x * beta[2]
}
```

```
fitted.values <- model1(c(50, 1.5), reg_data)
```

```
head(fitted.values)
```

```
## [1] 360.4925 415.7135 409.4930  73.7855  84.3965  51.2975
```

Let's now get the residuals of our model.

```
measure_distance <- function(mod, data) {
 diff <- data$y - model1(mod, data)
 sqrt(mean(diff ^ 2))
}
measure_distance(c(50, 1.5), sim1)
```

```
## [1] 42.828
```

We can use "purrr" to compute the distance for all the models defined previously. We will need a helper function because our distance expectes the model as a numeric vector of length 2.

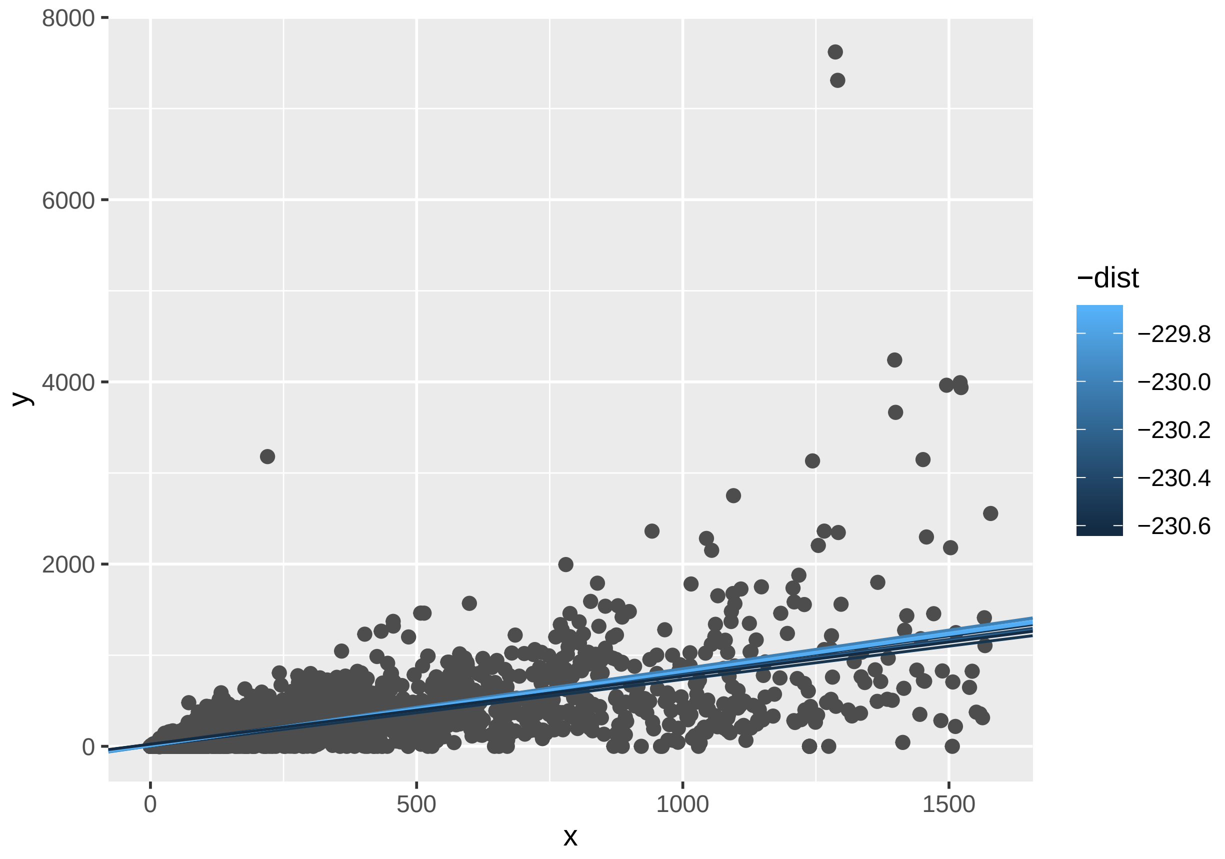
```
reg_data_dist <- function(beta1, beta2) {
 measure_distance(c(beta1, beta2), reg_data)
}
models <- models %>%
 mutate(dist = purrr::map2_dbl(beta1, beta2, reg_data_dist))
```

```
models
```

```
## # A tibble: 5,453 x 3
##      beta1  beta2  dist
##      <dbl>  <dbl> <dbl>
##   1 187.    3.60  1001.
##   2 140.   -1.55   707.
##   3 130.   -2.87  1100.
##   4   8.63 -0.253  397.
##   5 103.    1.75   425.
##   6  23.7  -3.42  1317.
##   7 153.    1.56   410.
##   8   3.18  0.731  231.
##   9 142.   -1.04   563.
##  10  81.7   3.75   983.
## # ... with 5,443 more rows
```

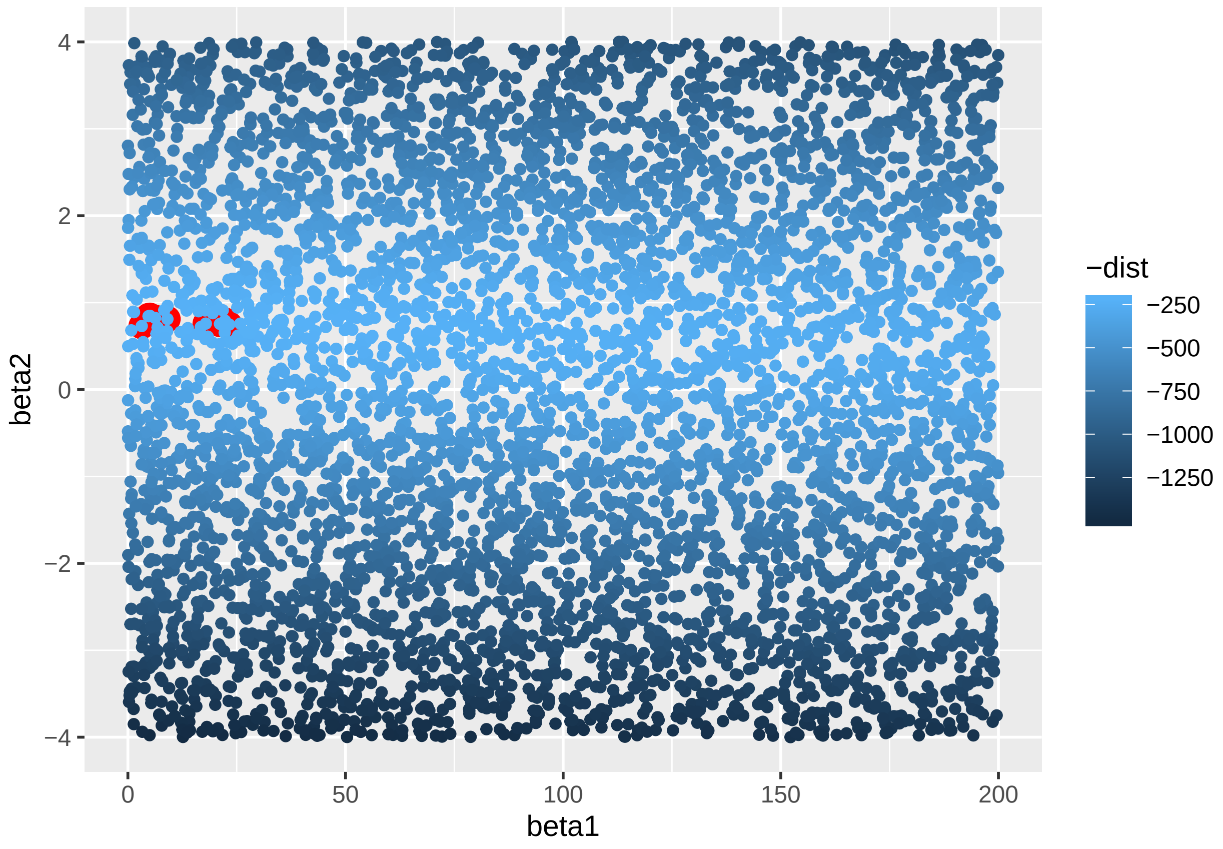
We can now overlay the best 10 models on the data.

```
ggplot(reg_data, aes(x, y)) +
 geom_point(size = 2, color = "grey30") +
 geom_abline(
 aes(intercept = beta1, slope = beta2, color = -dist),
 data = filter(models, rank(dist) <= 10)
 )
```



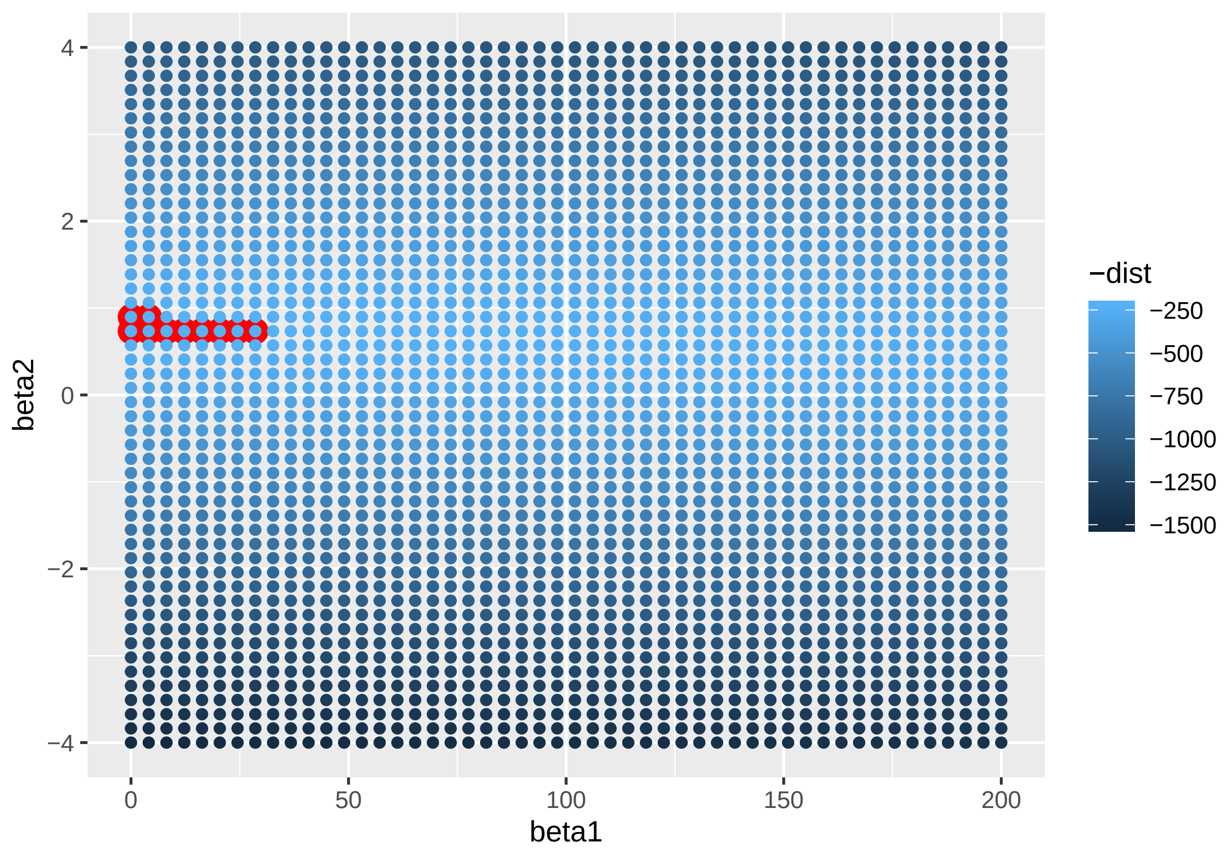
We can also think about these models as observations, and visualize them with a scatterplot of beta1 versus beta2, again colored by -dist. We can no longer directly see how the model compares to the data, but we can see many models at once. Again, I've highlighted the 10 best models, this time by drawing red circles underneath them:

```
ggplot(models, aes(beta1, beta2)) +
 geom_point(
 data = filter(models, rank(dist) <= 10),
 size = 4, color = "red"
 ) +
 geom_point(aes(colour = -dist))
```



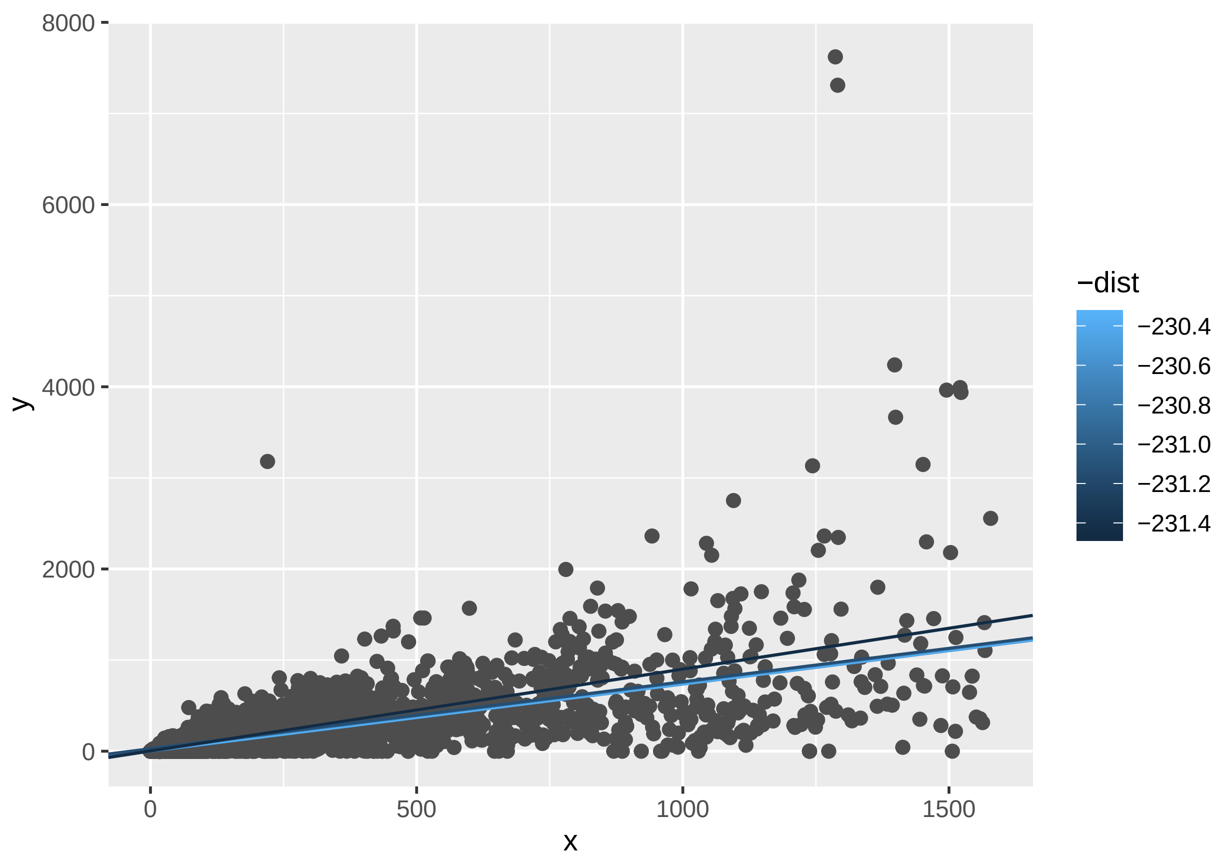
Instead of trying lots of random models, we could be more systematic and generate an evenly spaced grid of points (this is called a grid search). I picked the parameters of the grid roughly by looking at where the best models were in the preceding plot:

```
grid <- expand.grid(
 beta1 = seq(0, 200, length = 50),
 beta2 = seq(-4, 4, length = 50)
 ) %>%
 mutate(dist = purrr::map2_dbl(beta1, beta2, reg_data_dist))
grid %>%
 ggplot(aes(beta1, beta2)) +
 geom_point(
 data = filter(grid, rank(dist) <= 10),
 size = 4, colour = "red"
 ) +
 geom_point(aes(color = -dist))
```



When you overlay the best 10 models back on the original data, they all look pretty good:

```
ggplot(reg_data, aes(x, y)) +
 geom_point(size = 2, color = "grey30") +
 geom_abline(
 aes(intercept = beta1, slope = beta2, color = -dist),
 data = filter(grid, rank(dist) <= 10)
 )
```

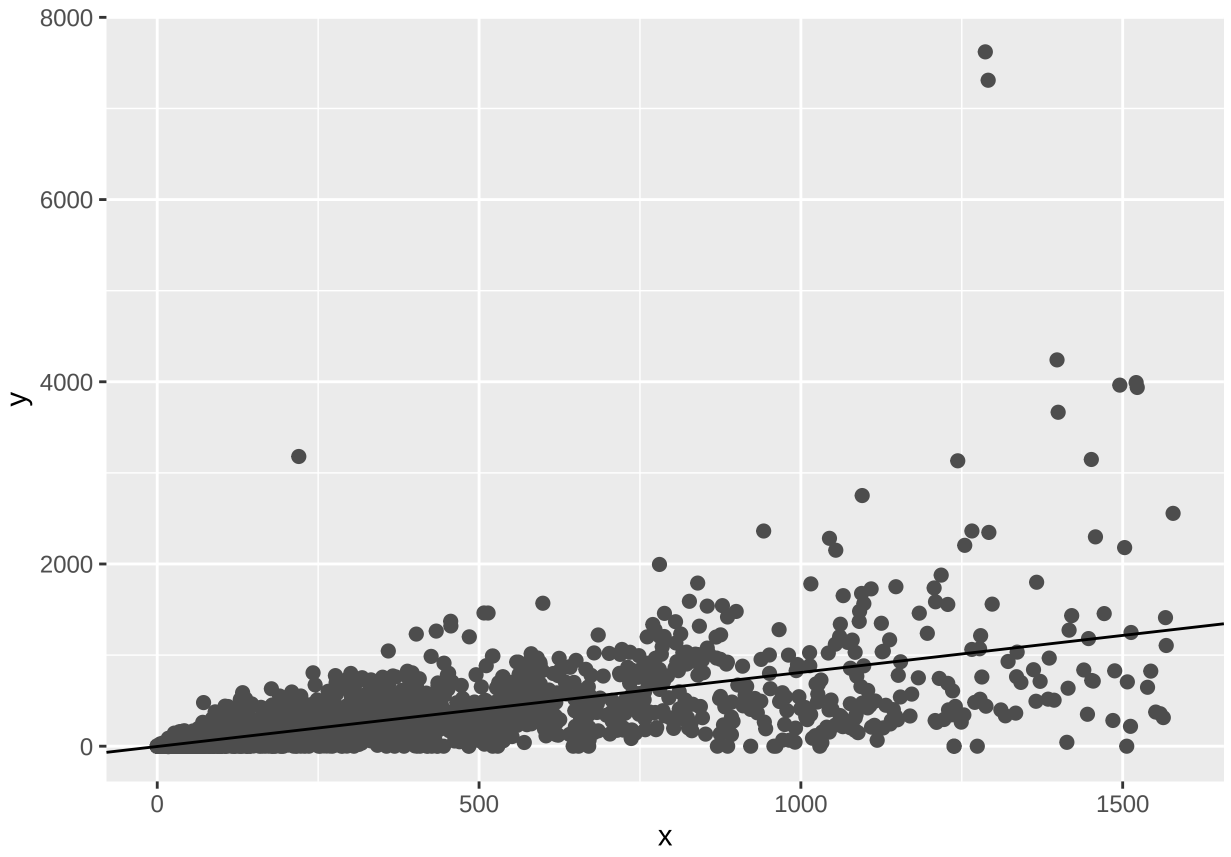


You could imagine iteratively making the grid finer and finer until you narrowed in on the best model. But there's a better way to tackle that problem: a numerical minimization tool called Newton-Raphson search. The intuition of Newton-Raphson is pretty simple: you pick a starting point and look around for the steepest slope. You then ski down that slope a little way, and then repeat again and again, until you can't go any lower. In R, we can do that with optim():

```
best <- optim(c(0, 0), measure_distance, data = reg_data)
best$par
```

```
## [1] -3.1063985  0.8127066
```

```
ggplot(reg_data, aes(x, y)) +
 geom_point(size = 2, color = "grey30") +
 geom_abline(intercept = best$par[1], slope = best$par[2])
```



Don't worry too much about the details of how optim() works. It's the intuition that's important here. If you have a function that defines the distance between a model and a dataset, and an algorithm that can minimize that distance by modifying the parameters of the model, you can find the best model. The neat thing about this approach is that it will work for any family of models that you can write an equation for. There's one more approach that we can use for this model, because it is a special case of a broader family: linear models. A linear model has the general form $y = a_1 + a_2 \cdot x_1 + a_3 \cdot x_2 + ... + a_n \cdot x_{(n-1)}$. So this simple model is equivalent to a general linear model where n is 2 and x_1 is x. R has a tool specifically designed for fitting linear models called lm(). lm() has a special way to specify the model family: formulas. Formulas look like y x, which lm() will translate to a function like $y = a_1 + a_2 * x$. We can fit the model and look at the output:

```
model_1 <- lm(y ~ x, data = reg_data)
summary(model_1)
```

```
##
## Call:
## lm(formula = y ~ x, data = reg_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1221.1   -27.4     0.9     9.9  6578.9
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.06484    3.62636  -0.845    0.398
## x            0.81267    0.01172  69.333   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 229.5 on 5451 degrees of freedom
## Multiple R-squared:  0.4686, Adjusted R-squared:  0.4685
## F-statistic:  4807 on 1 and 5451 DF,  p-value: < 2.2e-16
```

Now let's add an additional variable in the linear regression to compare the two different models.

```
z <- data_clean$Employees[which(data_clean$`Assets - Total`<
                          quantile(data_clean$`Assets - Total`, 0.95))]
reg_data <- cbind(reg_data, z)
```

```
model_2 <- lm(y ~ x + z, data = reg_data)
summary(model_2)
```

```
##
## Call:
## lm(formula = y ~ x + z, data = reg_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2675.1   -22.5     2.5    11.0  6700.1
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.31960    3.38664  -1.275    0.202
## x            0.69931    0.01166  59.999   <2e-16 ***
## z           20.62225    0.72862  28.303   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 214.3 on 5450 degrees of freedom
## Multiple R-squared:  0.5367, Adjusted R-squared:  0.5365
## F-statistic:  3157 on 2 and 5450 DF,  p-value: < 2.2e-16
```

In R, you can either write down all the variables that you want to use as regressors in your model or you can just use $y \sim .$.

```
model_3 <- lm(y ~ ., data = reg_data)
summary(model_3)
```

```
##
## Call:
## lm(formula = y ~ ., data = reg_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2675.1   -22.5     2.5    11.0  6700.1
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.31960    3.38664  -1.275    0.202
## x            0.69931    0.01166  59.999   <2e-16 ***
## z           20.62225    0.72862  28.303   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 214.3 on 5450 degrees of freedom
## Multiple R-squared:  0.5367, Adjusted R-squared:  0.5365
## F-statistic:  3157 on 2 and 5450 DF,  p-value: < 2.2e-16
```

A very easy way to compare two different linear regressions is through the likelihood ratio test. In statistics, the likelihood-ratio test assesses the goodness of fit of two competing statistical models based on the ratio of their likelihoods.

```
library(lmtest)
lrtest(model_1, model_2)
```

```
## Likelihood ratio test
##
## Model 1: y ~ x
## Model 2: y ~ x + z
##   #Df LogLik Df  Chisq Pr(>Chisq)
## 1   3 -37377
## 2   4 -37004  1 747.81  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

$p < 0.001$ indicates that the model with all predictors fits significantly better than the model with only one predictor. Another "goodness-of-fit" measure that can be used is the R^2:

$$R^2 = 1 - \frac{ESS}{TSS}. \tag{1}$$

```
summary(model_1)$r.squared
```

```
## [1] 0.4686107
```

```
summary(model_2)$r.squared
```

```
## [1] 0.5367084
```

We can also get the fitted values of the model for any x and z by running the following chunck of code.

```
coeffs = coefficients(model_2)
assets = 159
employees = 2
y <- coeffs[1] +coeffs[2]*assets +coeffs[3]*employees
y
```

```
## (Intercept)
##    148.1153
```

Or, equivalently:

```
newdata <- data.frame(x = 159, z = 2)
predict(model_2, newdata)
```

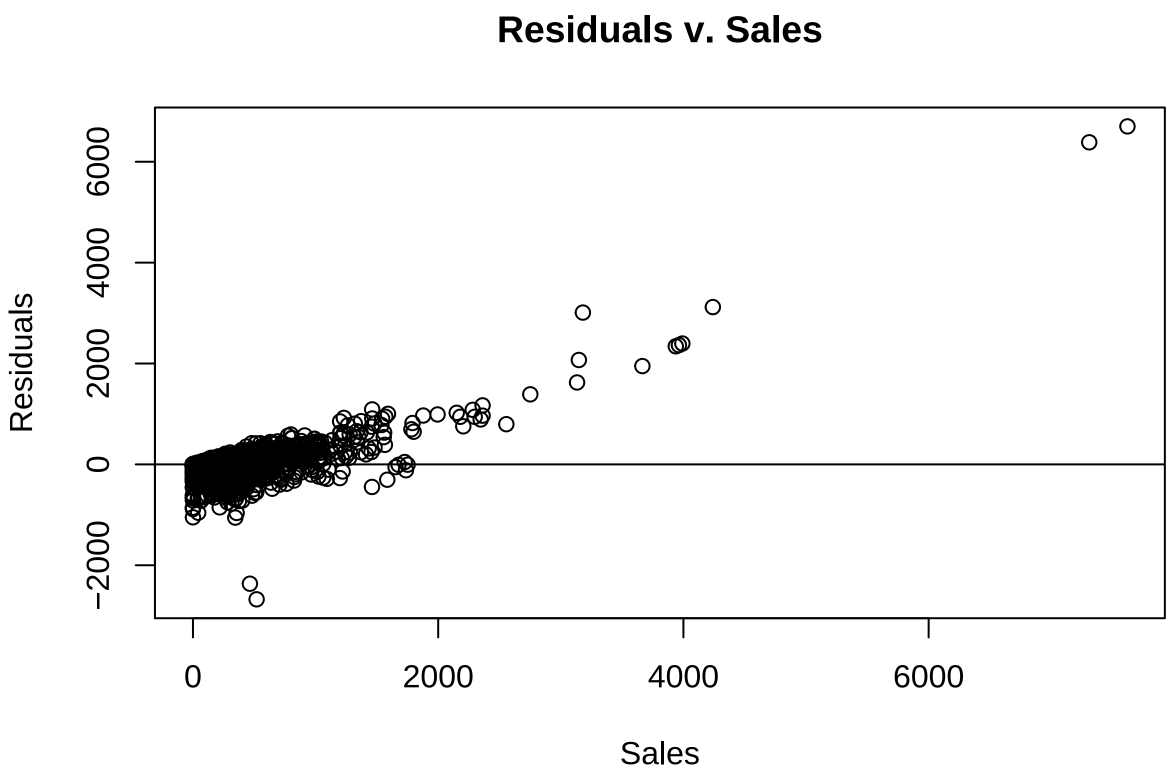
```
##        1
## 148.1153
```

```
predict(model_2, newdata, interval="confidence")
```

```
##        fit      lwr      upr
## 1 148.1153 142.2269 154.0036
```

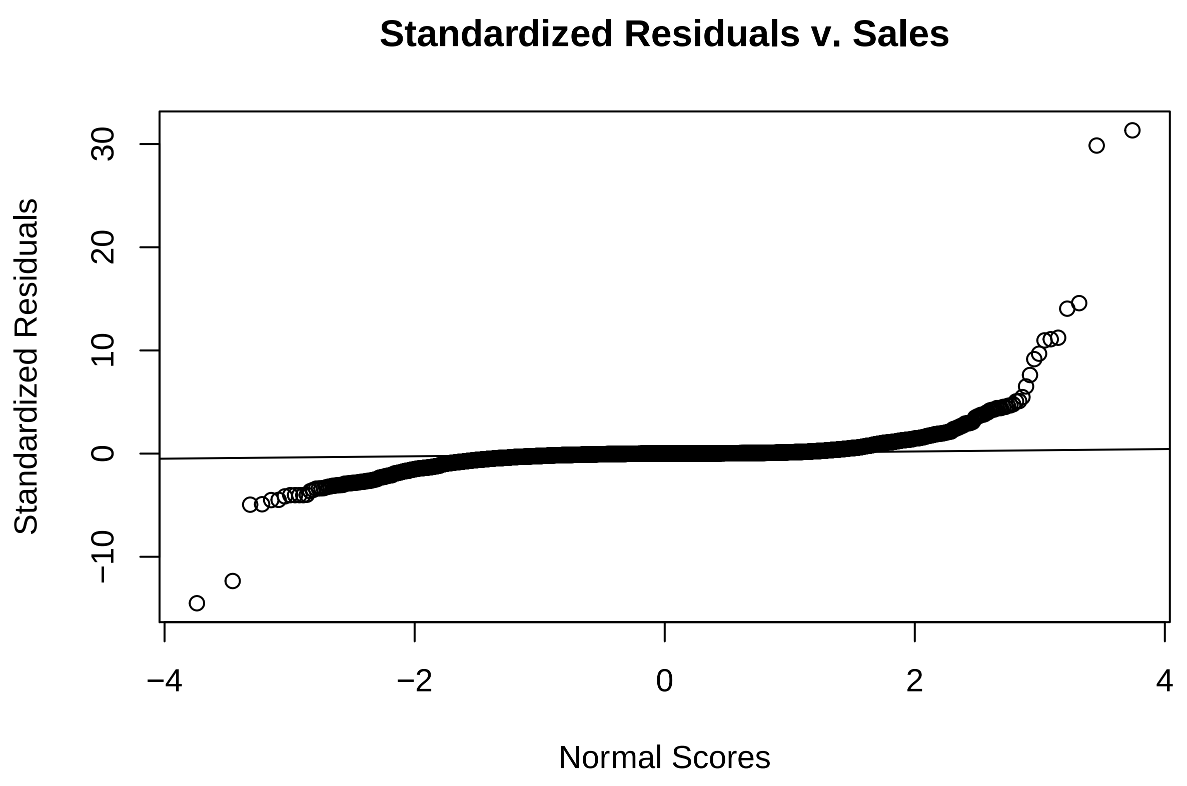
Once we fitted our favourite model, we can check the residuals from the model: $e_i = y_i - \hat{f}(x_i)$.

```
model.res = resid(model_2)
plot(reg_data$y, model.res, ylab="Residuals", xlab="Sales", main="Residuals v. Sales")
abline(0, 0)
```



Moreover, we can standardize the residuals and plot them against normalized scores for the outcome variable.

```
model_2.stdres = rstandard(model_2)
qqnorm(model_2.stdres ,  ylab="Standardized Residuals",  xlab="Normal Scores",  main="Standardized Resi
qqline(model_2.stdres)
```



In R, you can introduce an interaction between the regressors by using $*$. Always remember to include also the single regressors in the formula.

```
model_int<-lm(y ~ x  + z + x*z, data = reg_data)
summary(model_int)
```

```
##
## Call:
## lm(formula = y ~ x + z + x * z, data = reg_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1568.2   -25.4    -9.4     6.9  6775.0
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 9.308146   3.274783   2.842  0.00449 **
## x           0.605046   0.011784  51.343  < 2e-16 ***
## z           2.426270   1.033420   2.348  0.01892 *
## x:z         0.034463   0.001451  23.754  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 204 on 5449 degrees of freedom
## Multiple R-squared:  0.5802, Adjusted R-squared:  0.5799
## F-statistic:  2510 on 3 and 5449 DF,  p-value: < 2.2e-16
```

You can't directly introduce a quadratic term in the regression formula. Hence, you need to create an additional variable with the square term and then you can include it in the regression.

```
x2 <- x^2
model_squared<-lm(y ~ x  +  x2 + z, data = reg_data)
summary(model_squared)
```

```
##
## Call:
## lm(formula = y ~ x + x2 + z, data = reg_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2676.7   -22.3    -1.9     8.9  6669.9
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.055e+00  3.866e+00   0.273  0.78489
## x           6.189e-01  3.031e-02  20.418  < 2e-16 ***
## x2          7.753e-05  2.696e-05   2.875  0.00405 **
## z           2.066e+01  7.283e-01  28.370  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 214.1 on 5449 degrees of freedom
## Multiple R-squared:  0.5374, Adjusted R-squared:  0.5372
## F-statistic:  2110 on 3 and 5449 DF,  p-value: < 2.2e-16
```

Variables Selection

Here, I am going to show an application based based on an article from Barro and Lee (1994). The hypothesis we want to test is if less developed countries, with lower GDP per capita, grow faster than developed countries. In other words, there is a catch up effect. The model equation is as follows:

$$y_i = \alpha_0 d_i + \sum_{i=1}^{p} \beta_j x_{i,j} + \varepsilon_i \qquad (2)$$

where y_i is the GDP growth rate over a specific decade in country i, d_i is the log of the GDP at the beginning of the decade, $x_{i,j}$ are controls that may affect the GDP. We want to know the effects of d_i on y_i, which is measured by α_0. If our catch up hypothesis is true, α_0 must be positive and hopefully significant.

The dataset is available in the package. It has 62 variables and 90 observations. Each observation is a country, but the same country may have more than one observation if analysed in two different decades. The large number of variables will require some variable selection, and I will show what happens if we use a single LASSO selection and the Double Selection. The hdm package does all the DS steps in a single line of code, we do not need to estimate the two selection models and the Post-OLS individually. I will also run a naive OLS will all variables just for illustration. This application can be found here.

```
rm(list=ls())
data("GrowthData") # = use ?GrowthData for more information = #
dataset <- GrowthData[,-2] # = The second column is just a vector of ones = #
```

```
# = Naive OLS with all variables = #
# = I will select only the summary line that contains the initial log GDP = #
summary(lm(Outcome ~., data = dataset))
```

```
##
## Call:
## lm(formula = Outcome ~ ., data = dataset)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.040338 -0.011298 -0.000863  0.011813  0.043247
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.472e-01  7.845e-01   0.315  0.75506
## gdpsh465    -9.378e-03  2.989e-02  -0.314  0.75602
## bmp1l       -6.886e-02  3.253e-02  -2.117  0.04329 *
## freeop       8.007e-02  2.079e-01   0.385  0.70300
## freetar     -4.890e-01  4.182e-01  -1.169  0.25214
## h65         -2.362e+00  8.573e-01  -2.755  0.01019 *
## hm65         7.071e-01  5.231e-01   1.352  0.18729
## hf65         1.693e+00  5.032e-01   3.365  0.00223 **
## p65          2.655e-01  1.643e-01   1.616  0.11727
## pm65         1.370e-01  1.512e-01   0.906  0.37284
## pf65        -3.313e-01  1.651e-01  -2.006  0.05458 .
## s65          3.908e-02  1.855e-01   0.211  0.83469
## sm65        -3.067e-02  1.168e-01  -0.263  0.79479
## sf65        -1.799e-01  1.181e-01  -1.523  0.13886
## fert65       6.881e-03  2.705e-02   0.254  0.80108
## mort65      -2.335e-01  8.174e-01  -0.286  0.77729
## lifee065    -1.491e-02  1.933e-01  -0.077  0.93906
## gpop1        9.702e-01  1.812e+00   0.535  0.59663
## fert1        8.838e-03  3.504e-02   0.252  0.80271
## mort1        6.656e-02  6.848e-01   0.097  0.92326
## invsh41      7.446e-02  1.084e-01   0.687  0.49797
## geetot1     -7.151e-01  1.680e+00  -0.426  0.67364
## geerec1      6.300e-01  2.447e+00   0.257  0.79874
## gde1        -4.436e-01  1.671e+00  -0.265  0.79263
## govwb1       3.375e-01  4.380e-01   0.770  0.44748
## govsh41      4.632e-01  1.925e+00   0.241  0.81165
## gvxdxe41    -7.934e-01  2.059e+00  -0.385  0.70296
## high65      -7.525e-01  9.057e-01  -0.831  0.41311
## highm65     -3.903e-01  6.812e-01  -0.573  0.57131
## highf65     -4.177e-01  5.615e-01  -0.744  0.46308
## highc65     -2.216e+00  1.481e+00  -1.496  0.14575
## highcm65     2.797e-01  6.582e-01   0.425  0.67412
## highcf65     3.921e-01  7.660e-01   0.512  0.61278
## human65      2.337e+00  3.307e+00   0.707  0.48559
## humanm65    -1.209e+00  1.619e+00  -0.747  0.46121
## humanf65    -1.104e+00  1.685e+00  -0.655  0.51763
## hyr65        5.491e+01  2.389e+01   2.299  0.02918 *
## hyrm65       1.294e+01  2.317e+01   0.558  0.58112
## hyrf65       9.093e+00  1.767e+01   0.515  0.61088
## no65         3.721e-02  1.320e-01   0.282  0.78006
## nom65       -2.120e-02  6.496e-02  -0.326  0.74661
## nof65       -1.686e-02  6.700e-02  -0.252  0.80319
## pinstab1    -4.997e-02  3.092e-02  -1.616  0.11729
## pop65        1.032e-07  1.318e-07   0.783  0.44027
## worker65     3.408e-02  1.562e-01   0.218  0.82887
## pop1565     -4.655e-01  4.713e-01  -0.988  0.33176
## pop6565     -1.357e+00  6.349e-01  -2.138  0.04139 *
## sec65       -1.089e-02  3.077e-01  -0.035  0.97201
## secm65       3.344e-03  1.512e-01   0.022  0.98251
## secf65      -2.304e-03  1.580e-01  -0.015  0.98847
## secc65      -4.915e-01  7.290e-01  -0.674  0.50570
## seccm65      2.596e-01  3.557e-01   0.730  0.47150
## seccf65      2.207e-01  3.733e-01   0.591  0.55924
## syr65       -7.556e-01  7.977e+00  -0.095  0.92521
## syrm65       3.109e-01  3.897e+00   0.080  0.93698
## syrf65       7.593e-01  4.111e+00   0.185  0.85479
## teapri65     3.955e-05  7.700e-04   0.051  0.95941
## teasec65     2.497e-04  1.171e-03   0.213  0.83274
## ex1         -5.804e-01  2.418e-01  -2.400  0.02329 *
## im1          5.914e-01  2.503e-01   2.363  0.02531 *
## xr65        -1.038e-04  5.417e-05  -1.916  0.06565 .
## tot1        -1.279e-01  1.126e-01  -1.136  0.26561
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03074 on 28 degrees of freedom
## Multiple R-squared:  0.8871, Adjusted R-squared:  0.6411
## F-statistic: 3.607 on 61 and 28 DF,  p-value: 0.0002003
```

```
OLS <- summary(lm(Outcome ~., data = dataset))$coefficients[1, ]
OLS
```

```
##   Estimate Std. Error    t value   Pr(>|t|)
##  0.2471609  0.7845016  0.3150547  0.7550562
```

```
rlasso(Outcome~., data = dataset, post = FALSE)
```

```
##
## Call:
## rlasso.formula(formula = Outcome ~ ., data = dataset, post = FALSE)
##
## Coefficients:
## (Intercept)     gdpsh465        bmp1l       freeop      freetar
##   5.621e-02    0.000e+00    0.000e+00    7.020e-03   -1.748e-02
##         h65         hm65         hf65          p65         pm65
##   0.000e+00    0.000e+00   -1.093e-02    0.000e+00    0.000e+00
##        pf65          s65         sm65         sf65       fert65
##   0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00
##      mort65     lifee065        gpop1        fert1        mort1
##   0.000e+00    0.000e+00    0.000e+00    0.000e+00   -1.016e-01
##     invsh41      geetot1      geerec1         gde1       govwb1
##   0.000e+00    0.000e+00   -1.418e-01    4.126e-02    0.000e+00
##     govsh41     gvxdxe41       high65      highm65      highf65
##   0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00
##     highc65     highcm65     highcf65      human65     humanm65
##   0.000e+00    0.000e+00   -7.060e-04    0.000e+00    0.000e+00
##    humanf65        hyr65       hyrm65       hyrf65         no65
##   0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00
##       nom65       nof65    pinstab1       pop65    worker65
##   0.000e+00   0.000e+00   0.000e+00   0.000e+00   0.000e+00
##     pop1565     pop6565       sec65       secm65      secf65
##   0.000e+00   0.000e+00   0.000e+00   0.000e+00   0.000e+00
##      secc65      seccm65     seccf65       syr65      syrm65
##   0.000e+00   1.843e-04   0.000e+00   0.000e+00   0.000e+00
##      syrf65     teapri65    teasec65         ex1         im1
##   0.000e+00   0.000e+00   0.000e+00   0.000e+00   0.000e+00
##        xr65        tot1
##   1.386e-05   0.000e+00
```

```
# = Single step selection LASSO and Post-OLS = #
# = I will select only the summary line that contains the initial log GDP = #
lasso <- rlasso(Outcome~., data = dataset, post = FALSE) # = Run the Rigorous LASSO = #
selected <- which(coef(lasso)[-c(1:2)] !=0) # = Select relevant variables = #
selected
```

```
##   freeop  freetar     hf65    mort1  geerec1     gde1 highcf65  seccm65
##        2        3        6       18       21       22       31       50
##     xr65
##       59
```

```
fm <- paste(c("Outcome ~ gdpsh465", names(selected)), collapse = "+")
SS <- summary(lm(fm, data = dataset))$coefficients[1, ]
SS
```

```
##    Estimate  Std. Error     t value    Pr(>|t|)
## 0.311687933 0.098324653 3.169987628 0.002169693
```

```
# = Double Selection = #
X <- as.matrix(dataset[,-1])
y <- dataset$Outcome
DS <- rlassoEffects(X , y, I = ~ dataset$gdpsh465, data = dataset)
DS <- summary(DS)$coefficients[1,]
```

```
results <- rbind(OLS,SS,DS)
results
```

```
##        Estimate Std. Error    t value     Pr(>|t|)
## OLS  0.24716089 0.78450163  0.3150547 0.7550561700
## SS   0.31168793 0.09832465  3.1699876 0.0021696930
## DS  -0.04981147 0.01393636 -3.5742095 0.0003512875
```

The OLS estimate is positive, however the standard error is very big because we have only 90 observations for more than 60 variables. The Single Selection estimate is also positive and, in this case, significant. However, the Double Selection showed a negative and significant coefficient. If the DS is correct, our initial catch up hypothesis is wrong and poor countries grow less than rich countries. We can't say that the DS is correct for sure, but it is backed up by a strong theory and lots of simulations that show that the SS is problematic. It is very, very unlikely that the SS results are more accurate than the DS. It is very surprising how much the results can change from one case to the other. You should at least be skeptic when you see this type of modelling and the selection of controls is not clear.

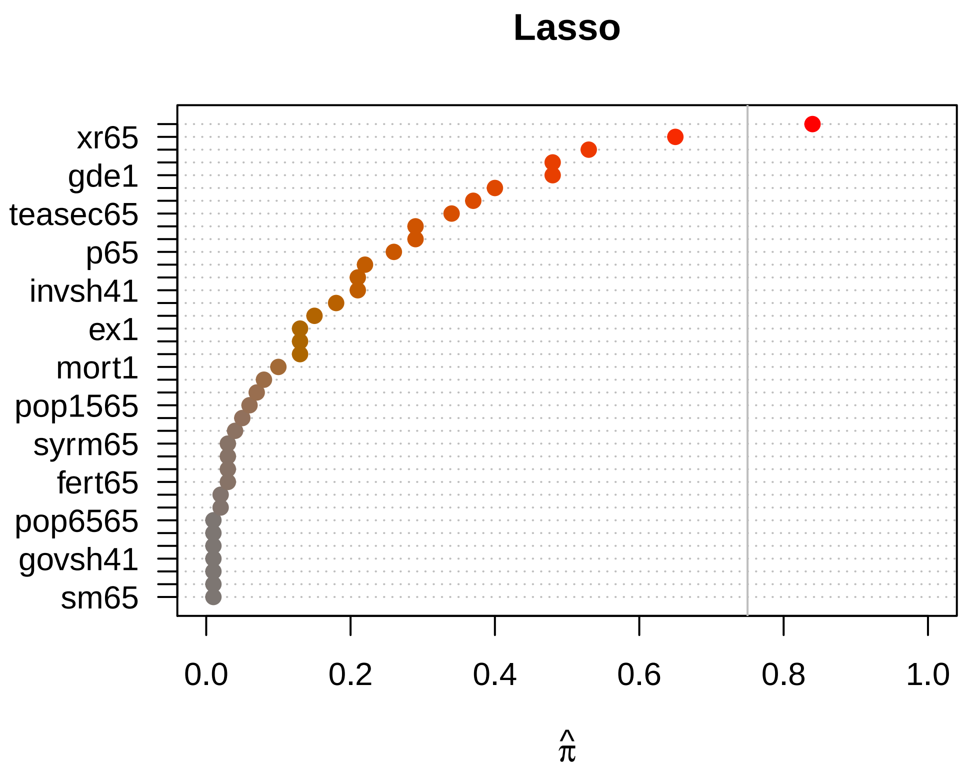
The "hdm" package has several other implementations in this framework such as instrumental variables and logit models and there are also more examples in the package vignette.

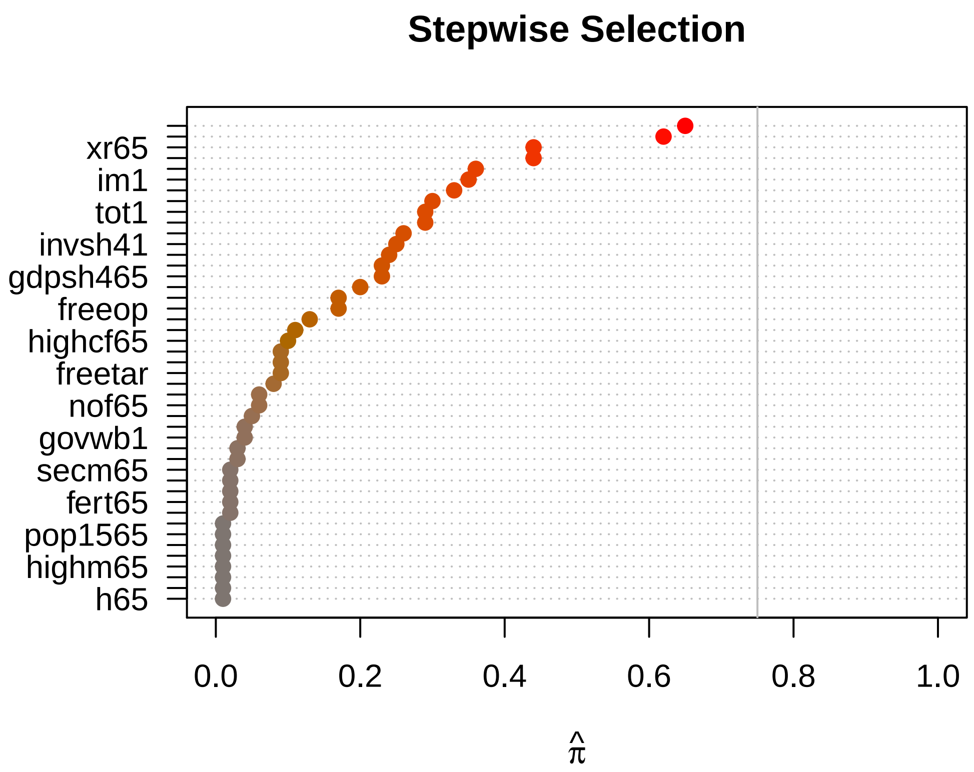
Stability Selection

```
###########################################################################
### using stability selection with Lasso methods:

if (require("lars")) {
    (stab.lasso <- stabsel(x = X, y = y,
                           fitfun = lars.lasso, cutoff = 0.75,
                           PFER = 1))
    (stab.stepwise <- stabsel(x = X, y = y,
                              fitfun = lars.stepwise, cutoff = 0.75,
                              PFER = 1))
    plot(stab.lasso, main = "Lasso")
    plot(stab.stepwise, main = "Stepwise Selection")
    ## --> stepwise selection seems to be quite unstable even in this low
    ##     dimensional example!
}
```

```
## Loading required package: lars
```

```
## Loaded lars 1.2
```

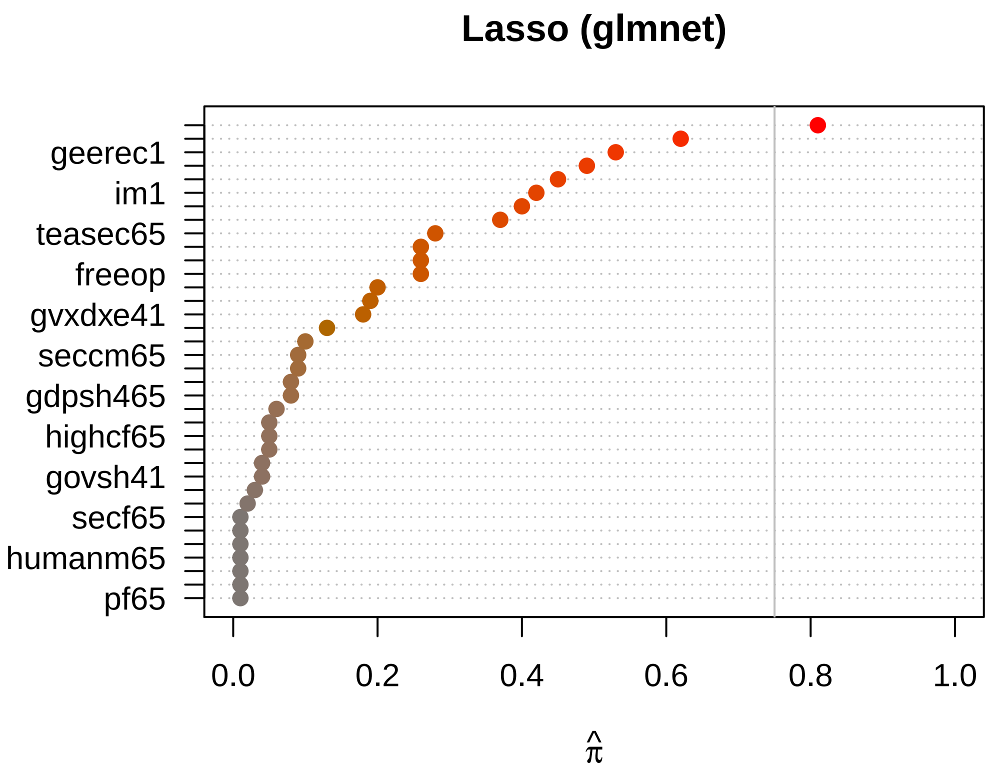


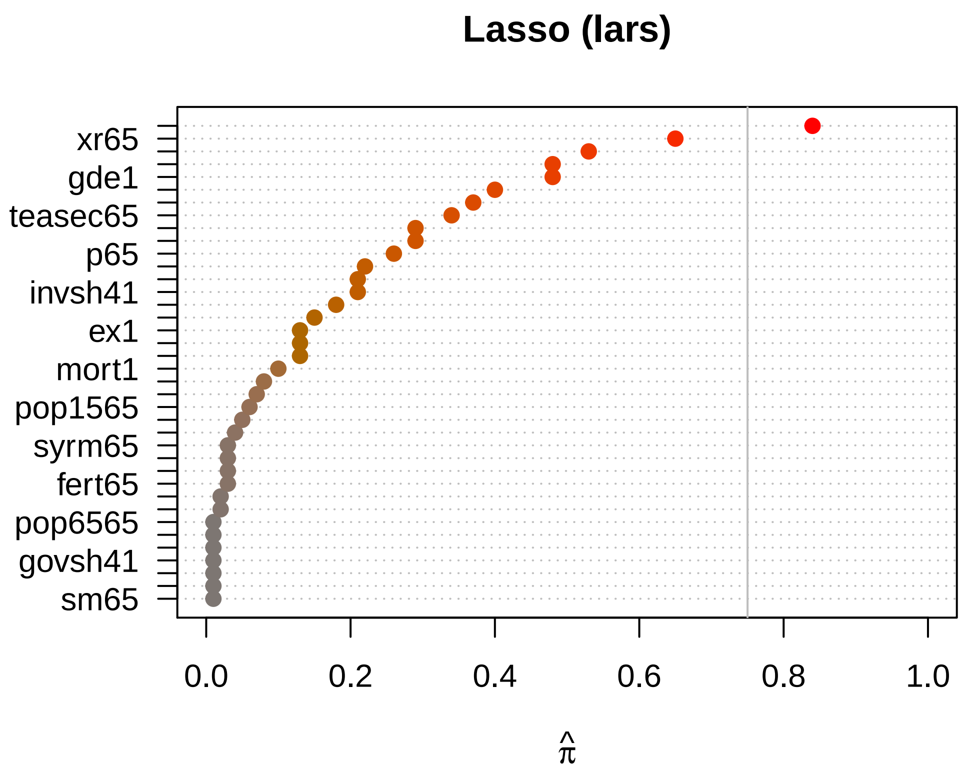


```r
## set seed (again to make results comparable)
set.seed(1234)
if (require("glmnet")) {
    (stab.glmnet <- stabsel(x = X, y = y,
                            fitfun = glmnet.lasso, cutoff = 0.75,
                            PFER = 1))
    plot(stab.glmnet, main = "Lasso (glmnet)")
    if (exists("stab.lasso"))
        plot(stab.lasso, main = "Lasso (lars)")
}
```

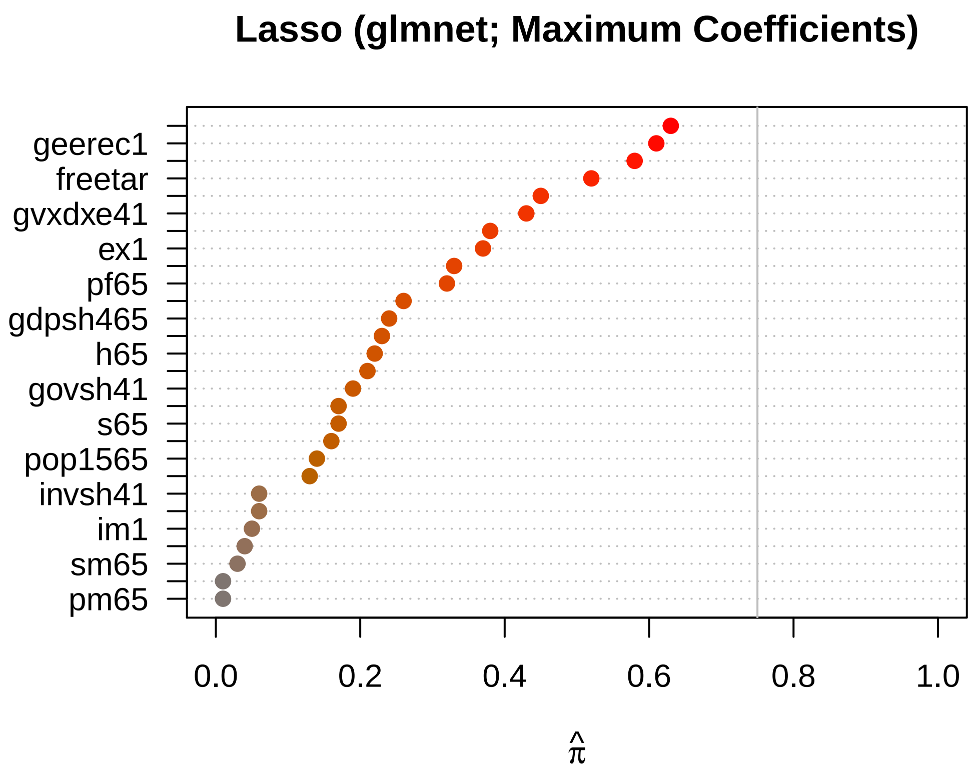
```
## Loading required package: glmnet

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loading required package: foreach

##
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':
##
```

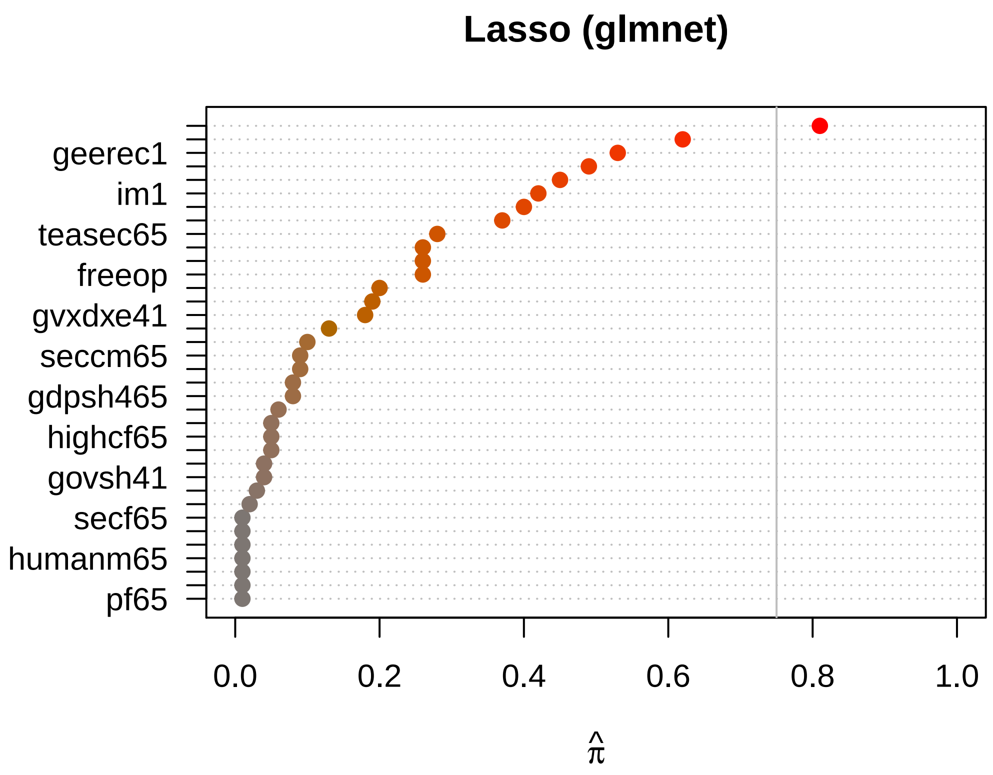
```
##     accumulate, when
## Loaded glmnet 2.0-18
```





```
## Select variables with maximum coefficients based on lasso estimate

set.seed(1234) # reset seed
if (require("glmnet")) {
    ## use cross-validated lambda
    lambda.min <- cv.glmnet(x = as.matrix(X), y = y)$lambda.min
    (stab.maxCoef <- stabsel(x = X, y = y,
                            fitfun = glmnet.lasso_maxCoef,
                            # specify additional parameters to fitfun
                            args.fitfun = list(lambda = lambda.min),
                            cutoff = 0.75, PFER = 1))

    ## WARNING: Using a fixed penalty (lambda) is usually not permitted and
    ##          not sensible. See ?fitfun for details.

    ## now compare standard lasso with "maximal parameter estimates" from lasso
    plot(stab.maxCoef, main = "Lasso (glmnet; Maximum Coefficients)")
    plot(stab.glmnet, main = "Lasso (glmnet)")
    ## --> very different results.
}
```


Lasso (glmnet; Maximum Coefficients)
geerec1
freetar
gvxdxe41
ex1
pf65
gdpsh465
h65
govsh41
s65
pop1565
invsh41
im1
sm65
pm65
0.0
0.2
0.4
0.6
0.8
1.0
$\hat{\pi}$



Instruments Selection

Reproduction of the analysis by Angrist and Krueger (1991).

```r
load("G:\\Il mio Drive\\Teaching\\Data Science Lab 2020\\angrist_krueger_1991.rda")
ak91 <- mutate(ak91,
              qob_fct = factor(qob),
              q4 = as.integer(qob == "4"),
              yob_fct = factor(yob))
```

Regress log wages on 4th quarter.

```r
mod1 <- lm(lnw ~ q4, data = ak91)
coeftest(mod1, vcov = sandwich)
```

```
##
## t test of coefficients:
##
##              Estimate Std. Error   t value Pr(>|t|)
## (Intercept) 5.8982723  0.0013625 4329.1303  < 2e-16 ***
## q4          0.0068132  0.0027433    2.4836  0.01301 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Regress years of schooling on 4th quarter.

```
mod2 <- lm(s ~ q4, data = ak91)
coeftest(mod2, vcov = sandwich)
```

```
##
## t test of coefficients:
##
##               Estimate Std. Error   t value  Pr(>|t|)
## (Intercept) 12.7473106  0.0066085 1928.9230 < 2.2e-16 ***
## q4           0.0921209  0.0131613    6.9994 2.576e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

IV regression of log wages on years of schooling, with 4th quarter as an instrument for years of schooling.

```
mod3 <- ivreg(lnw ~ s | q4, data = ak91)
coeftest(mod3, vcov = sandwich)
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept) 4.955495   0.357736 13.8524 < 2.2e-16 ***
## s           0.073959   0.028014  2.6401  0.008289 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Regression Estimates of Returns to Schooling using Quarter of Birth Instruments

```
mod4 <- lm(lnw ~ s, data = ak91)
coeftest(mod4, vcov = sandwich)
```

```
##
## t test of coefficients:
##
##               Estimate Std. Error t value  Pr(>|t|)
## (Intercept) 4.99518231 0.00507386  984.49 < 2.2e-16 ***
## s           0.07085104 0.00038102  185.95 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

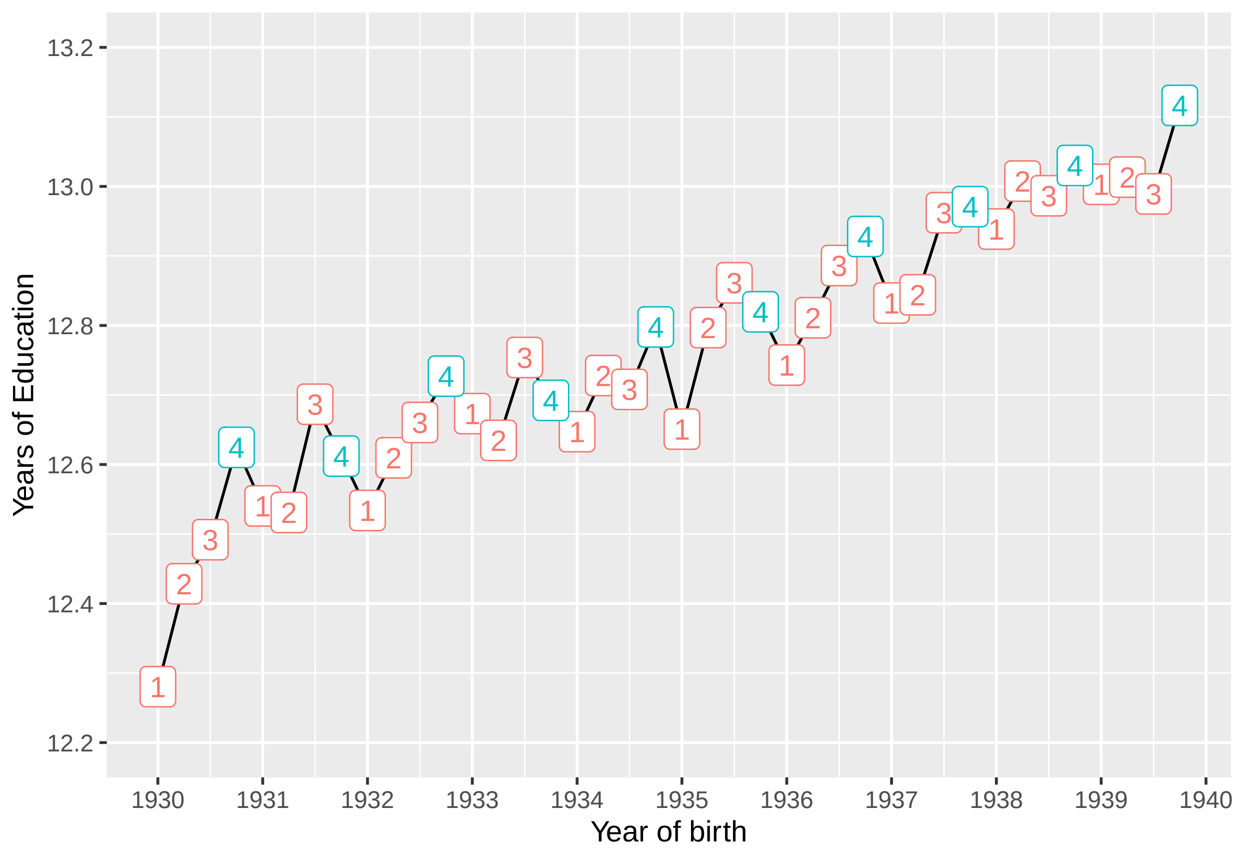
IV reg using all quarters as instruments. Controls for year of birth.

```
mod5 <- ivreg(lnw ~ s + yob_fct | qob_fct + yob_fct, data = ak91)
summary(mod5, vcov = sandwich, diagnostics = TRUE)
```

```
##
## Call:
## ivreg(formula = lnw ~ s + yob_fct | qob_fct + yob_fct, data = ak91)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.9945 -0.2544  0.0676  0.3509  4.8425
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.591739   0.250574  18.325  < 2e-16 ***
## s            0.105252   0.020115   5.232 1.67e-07 ***
## yob_fct1931 -0.011113   0.005906  -1.882 0.059878 .
## yob_fct1932 -0.020886   0.006227  -3.354 0.000796 ***
## yob_fct1933 -0.025565   0.006976  -3.665 0.000248 ***
## yob_fct1934 -0.030072   0.007421  -4.053 5.07e-05 ***
## yob_fct1935 -0.044136   0.008363  -5.278 1.31e-07 ***
## yob_fct1936 -0.045008   0.009301  -4.839 1.30e-06 ***
## yob_fct1937 -0.052070   0.010335  -5.038 4.70e-07 ***
## yob_fct1938 -0.055185   0.011840  -4.661 3.15e-06 ***
## yob_fct1939 -0.067800   0.012589  -5.386 7.22e-08 ***
##
## Diagnostic tests:
##                     df1    df2 statistic p-value
## Weak instruments      3 329496    32.325  <2e-16 ***
## Wu-Hausman            1 329497     2.977  0.0844 .
## Sargan                2     NA     3.259  0.1960
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6474 on 329498 degrees of freedom
## Multiple R-Squared: 0.09054, Adjusted R-squared: 0.09051
## Wald test: 3.793 on 10 and 329498 DF,  p-value: 3.898e-05
```

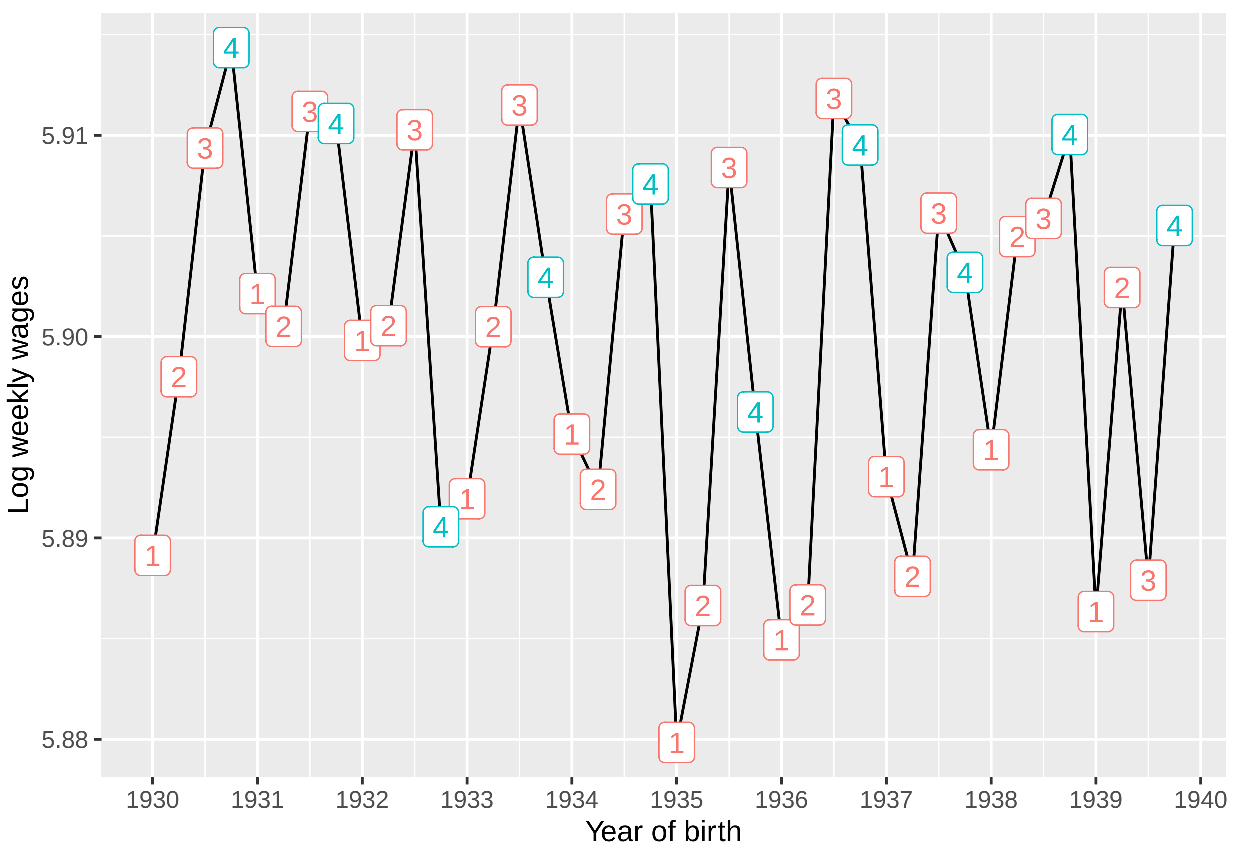
Average years of schooling by quarter of birth for men born in 1930-39 in the 1980 US Census.

```
ak91_age <- ak91 %>%
  group_by(qob, yob) %>%
  summarise(lnw = mean(lnw), s = mean(s)) %>%
  mutate(q4 = (qob == 4))
ggplot(ak91_age, aes(x = yob + (qob - 1) / 4, y = s)) +
  geom_line() +
  geom_label(mapping = aes(label = qob, color = q4)) +
  theme(legend.position = "none") +
  scale_x_continuous("Year of birth", breaks = 1930:1940) +
  scale_y_continuous("Years of Education", breaks = seq(12.2, 13.2, by = 0.2),
                     limits = c(12.2, 13.2))
```



Average log wages by quarter of birth for men born in 1930-39 in the 1980 US Census.

```r
ggplot(ak91_age, aes(x = yob + (qob - 1) / 4, y = lnw)) +
  geom_line() +
  geom_label(mapping = aes(label = qob, color = q4)) +
  scale_x_continuous("Year of birth", breaks = 1930:1940) +
  scale_y_continuous("Log weekly wages") +
  theme(legend.position = "none")
```



Log weekly wages
5.91
5.90
5.89
5.88
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
Year of birth