Biostatistics week 11

How to best use regression for prediction?

Topics of this lecture

- Purpose of descriptive vs. predictive regression models
- What data do we need for predictive modeling
- Rigid vs flexible models: Underfitting/Overfitting or Bias/Variance
- How to evaluate a predictive regression model: MSE on new data
- Shrinkage of parameter in linear models help to get better predictions
- Lasso or ridge regression yield good prediction models
- Regression to the mean and why this matters in medicine
- The best descriptive model is often not the best predictive model

For what purpose do we develop a statistical model?

Statistical Science
2010, Vol. 25, No. 3, 289–310
DOI: 10.1214/10-STS330

To Explain or to Predict?

Galit Shmueli

- **Description:**
  Describe data by a statistical model.

  ← Remember last lecture

- **Explanation:**
  Search for the "true" model to understand and causally explain the relationships between variables and to plan for interventions.

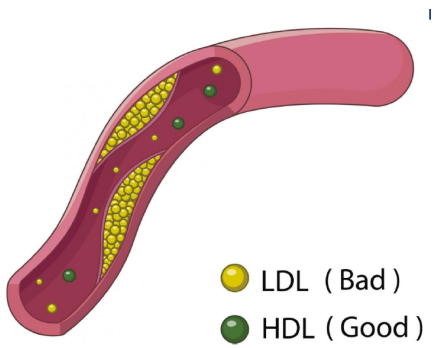
  ← Difficult with observational data – in medicine we do RCT to learn about causal effects

- **Prediction:**
  Use model to make reliable predictions.

  ← Main topic of today

Descriptive modelling: we saw that last time

	Estimate	Std. Error	t-value	Pr(>|t|)
Intercept	1.16448	0.28804	4.04	<.0001
AGE	-0.00092	0.00125	-0.74	0.4602
BMI	-0.01205	0.00295	-4.08	<.0001
BLC	0.05055	0.02215	2.28	0.0239
PRSSY	-0.00041	0.00044	-0.95	0.3436
DIAST	0.00255	0.00103	2.47	0.0147
GLUM	-0.00046	0.00018	-2.50	0.0135
SKINF	0.00147	0.00183	0.81	0.4221
LCHOL	0.31109	0.10936	2.84	0.0051



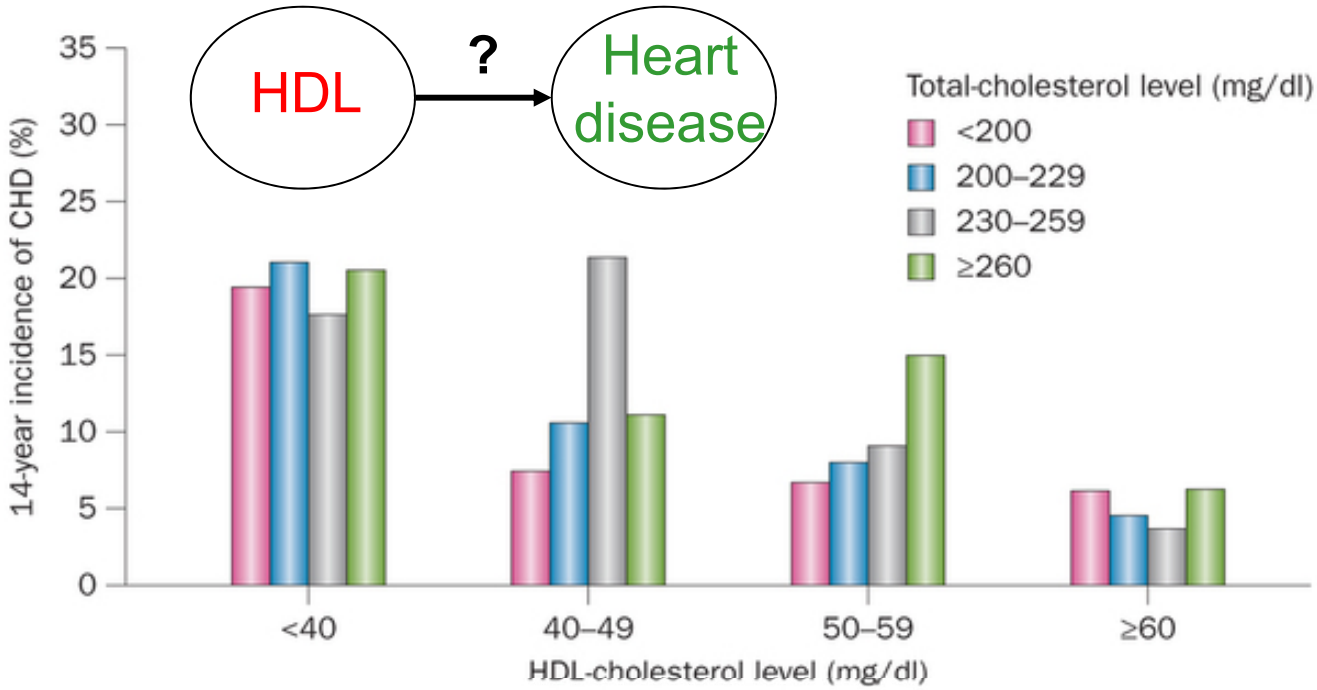
$$\log(HDL) = 1.16 - 0.00092 \cdot AGE - 0.01 \cdot BMI + 0.05 \cdot BLC + \cdots + 0.31 \cdot LCHOL$$

$$\hat{\beta}_3 = 0.05 = y_{x_3+1} - y_{x_3}$$

The coefficient β_3 gives the change of the outcome y=log(HDL), given the explanatory variable BLC_k is (vitamin in blood) is increased by one unit and **all other variables are hold constant** (usually not a realistic assumption).

Bad news: you can only estimate, but never “observe directly” the coefficient of a model.

A warning ahead:
A good predictive model may not be a good causal model



Epidemiological studies of CHD and the evolution of preventive cardiology Nature Reviews Cardiology 11, 276–289 (2014)

HDL gives a strong negative association with heart disease in cross-sectional studies and is the strongest predictor of future events in prospective studies.

Prediction worked

Roche tested the effect of drug "dalcetrapib" in phase III on 15'000 patients which proved to boost HDL ("good cholesterol") but failed to prevent heart diseases. Roche stopped the failed trial on May 2012 and immediately lost $5billion of its market capitalization.

Planned intervention failed

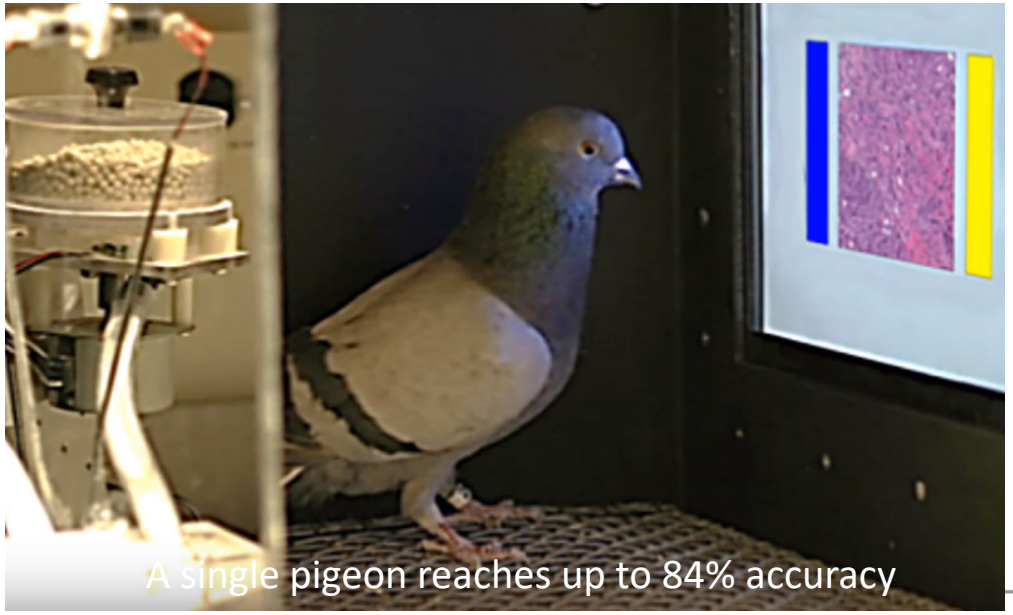
Prediction is often easier than explanation

I saw such guys before and hearing these numbers the blood pressure is Y

He has a LDL of X1 which narrows your blood vessels beside that with his age of X2 their flexibility is reduced, so that I deduce a blood pressure of Y

Prediction does not require understanding

https://www.youtube.com/watch?v=NsV6S8EsC0E



PLOS ONE

RESEARCH ARTICLE

Pigeons (*Columba livia*) as Trainable Observers of Pathology and Radiology Breast Cancer Images

Richard M. Levenson[1]*, Elizabeth A. Krupinski[3], Victor M. Navarro[2], Edward A. Wasserman[2]*

1 Department of Pathology and Laboratory Medicine, University of California Davis Medical Center, Sacramento, California, United States of America, **2** Department of Psychological and Brain Sciences, The University of Iowa, Iowa City, Iowa, United States of America, **3** Department of Radiology & Imaging Sciences, College of Medicine, Emory University, Atlanta, Georgia, United States of America

* levenson@ucdavis.edu (RML); ed-wasserman@uiowa.edu (EAW)

OPEN ACCESS

Citation: Levenson RM, Krupinski EA, Navarro VM, Wasserman EA (2015) Pigeons (*Columba livia*) as Trainable Observers of Pathology and Radiology Breast Cancer Images. PLoS ONE 10(11): e0141357. doi:10.1371/journal.pone.0141357

Editor: Jonathan A Coles, Glasgow University, UNITED KINGDOM

Received: August 25, 2015

Accepted: October 7, 2015

Published: November 18, 2015

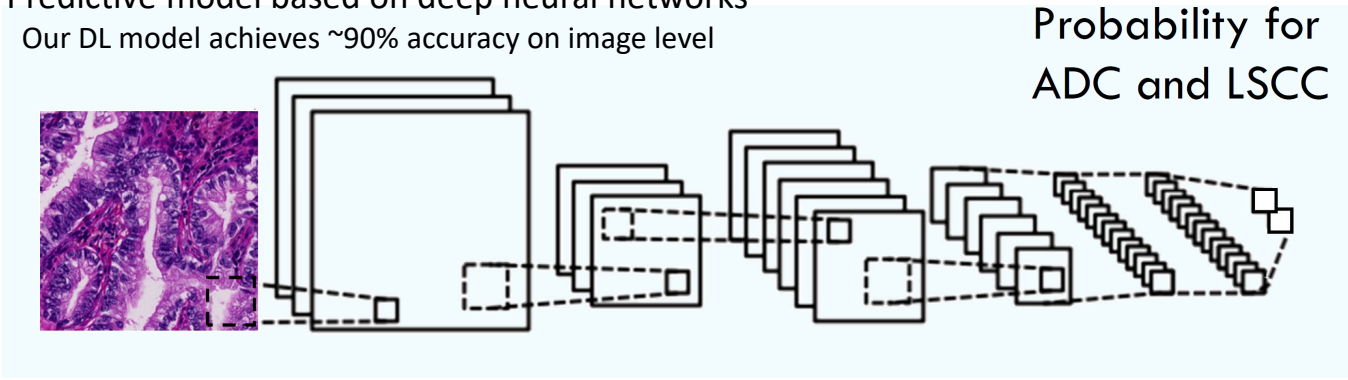
Abstract

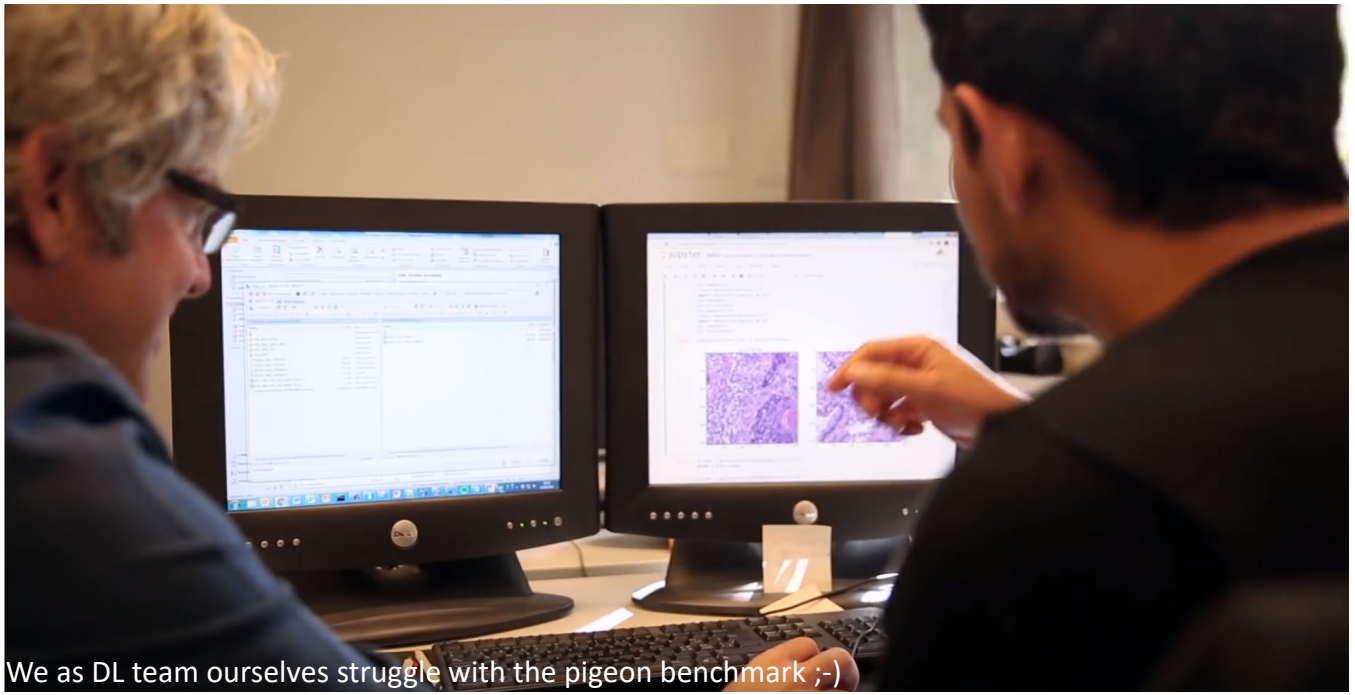
Pathologists and radiologists spend years acquiring and refining their medically essential visual skills, so it is of considerable interest to understand how this process actually unfolds and what image features and properties are critical for accurate diagnostic performance. Key insights into human behavioral tasks can often be obtained by using appropriate animal models. We report here that pigeons (*Columba livia*)—which share many visual system properties with humans—can serve as promising surrogate observers of medical images, a capability not previously documented. The birds proved to have a remarkable ability to distinguish benign from malignant human breast histopathology after training with differential food reinforcement; even more importantly, the pigeons were able to generalize what they had learned when confronted with novel image sets. The birds' histological accuracy, like that of humans, was modestly affected by the presence or absence of color as well as by degrees of image compression, but these impacts could be ameliorated with further training. Turning to radiology, the birds proved to be similarly capable of detecting cancer-relevant microcalcifications on mammogram images. However, when given a different (and for humans quite difficult) task—namely, classification of suspicious mammographic densities (masses)—the pigeons proved to be capable only of image memorization and were unable

Predictive models are still not always easy to beat

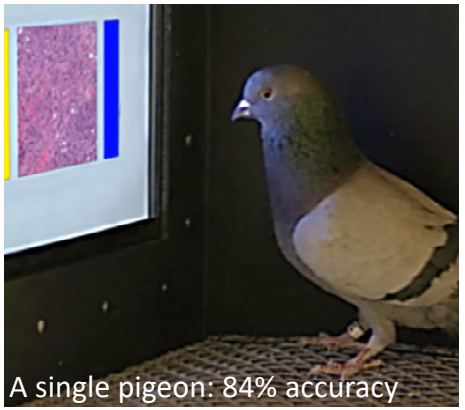
Predictive model based on deep neural networks

Our DL model achieves ~90% accuracy on image level





We as DL team ourselves struggle with the pigeon benchmark ;-)



A single pigeon: 84% accuracy

Predictive modeling

What data do we need to build a prediction model?

- We need to observe the outcome
- We should observe as many potential predictors as possible
- We should think about transforming the variables before fitting
- We should collect the outcome and predictor variables for a large, representative patient sample, so that ideally ~30% is enough to fit a model that contains all predictors of interest and also some reasonable interactions, e.g.:

simple model:

$$y = \beta_0 + \beta_1 \cdot x_1 + \varepsilon$$

complex model

$$\log(y) = \beta_0 + \beta_1 \cdot x_1^2 + \beta_2 \cdot \sqrt{x_1} + \beta_3 \cdot x_3 + \beta_4 \cdot \sqrt{x_1} \cdot x_2 + \varepsilon$$

Predictive modeling

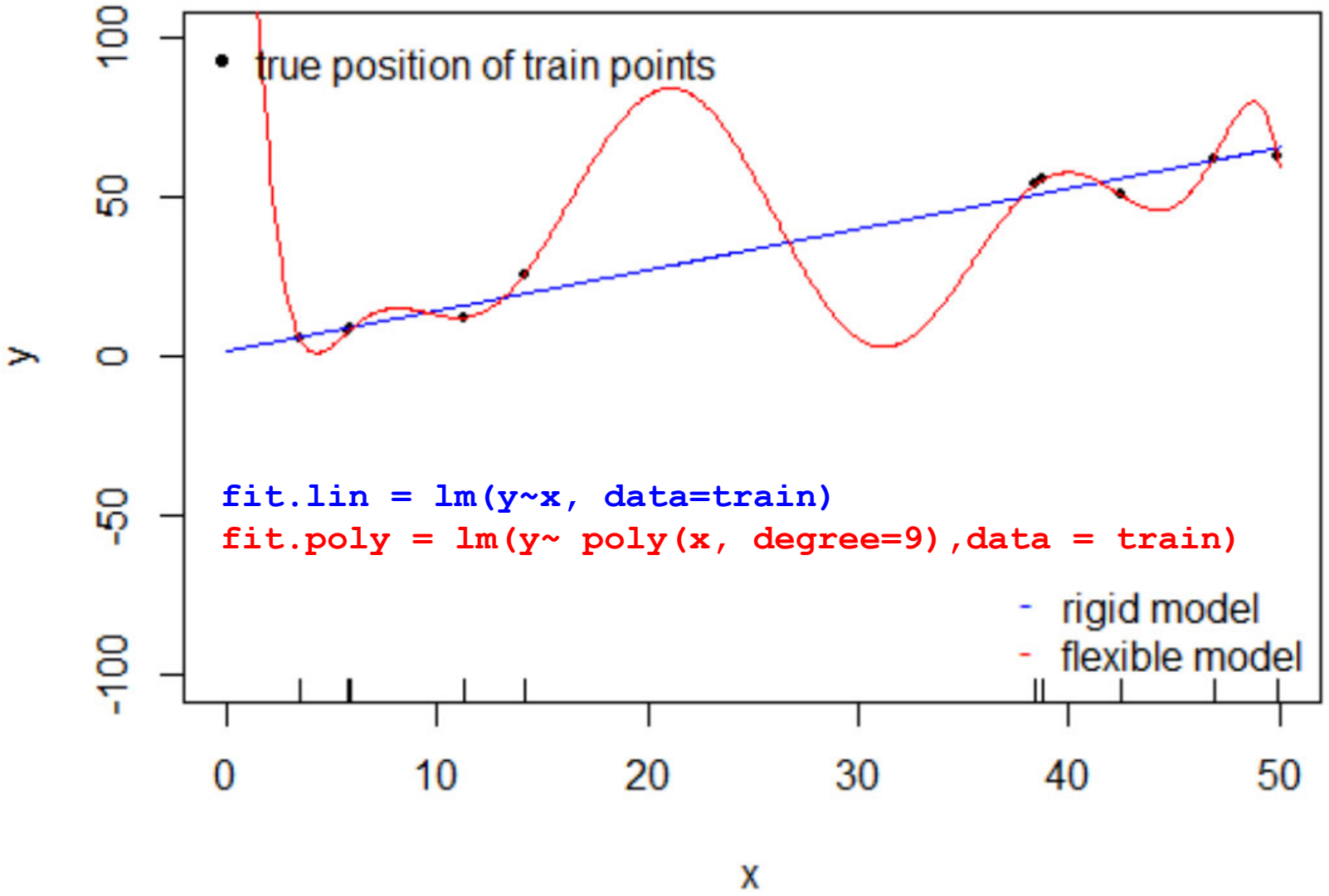
Nils Bohr, physics Nobel price 1922

What data do we need to evaluate a prediction model?

Always use new data, that were not used to build the model, to evaluate the predictive performance!

Good news: for predictive models we can eventually observe the true value
→ we can directly check how good our predictions are

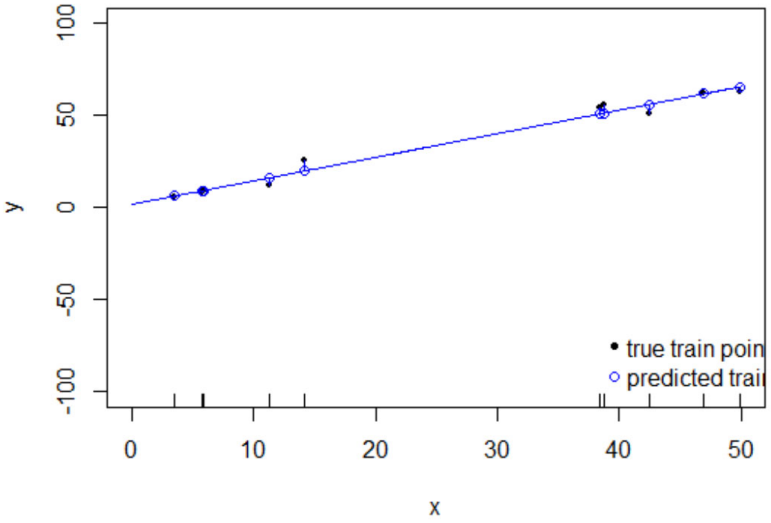
Which model will yield the better predictions?



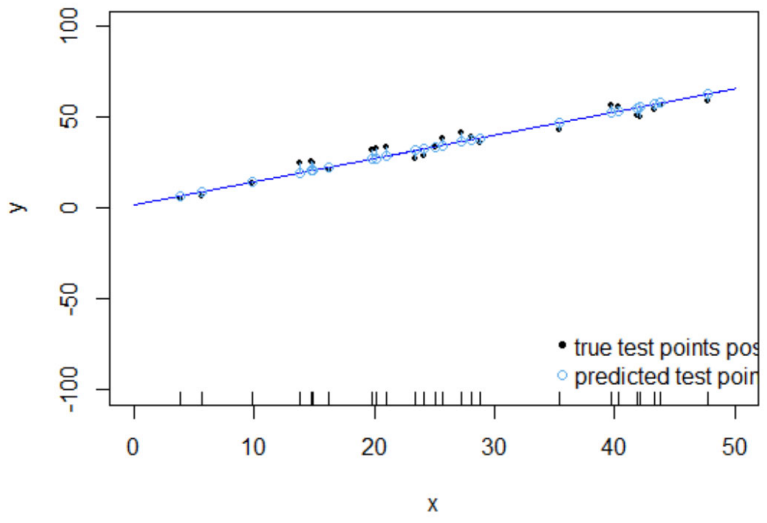
Since we simulate this data, we know the true relationship between x and y and can hence easily sample random training (and test) data which we can use to fit (and evaluate) a model.

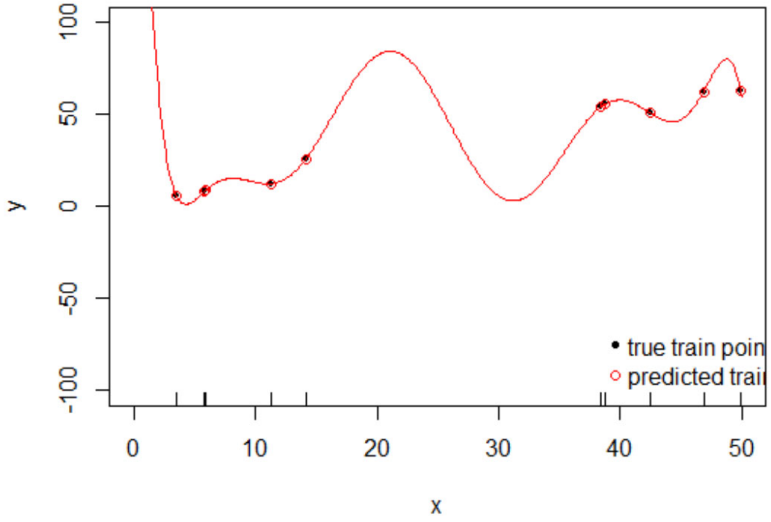
Compare flexible with ridge prediction model

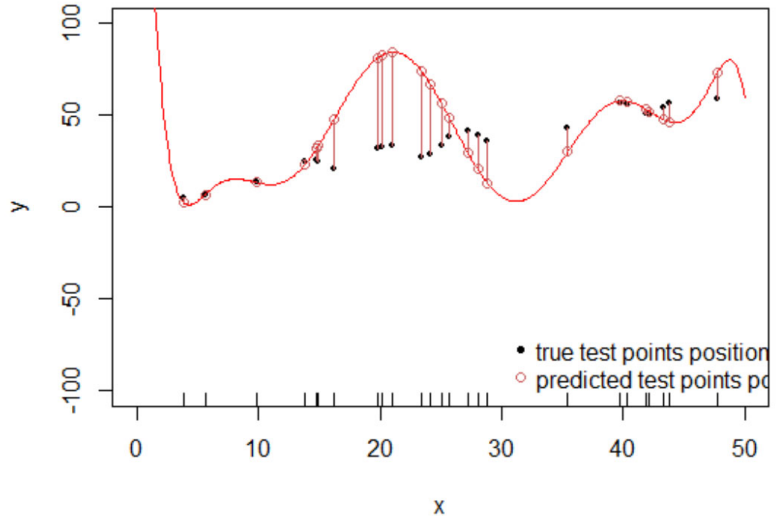
Check performance on **train** data



Check performance on **test** data





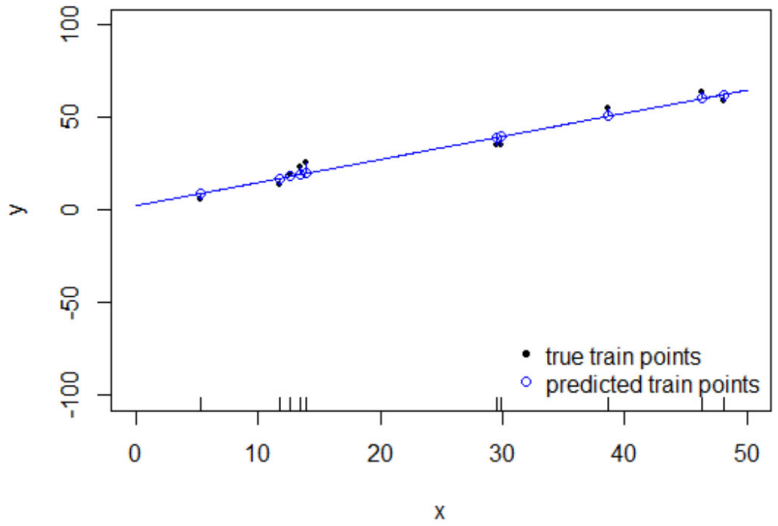


Note: The flexible “overfits” the train data: its performance goes down on test data

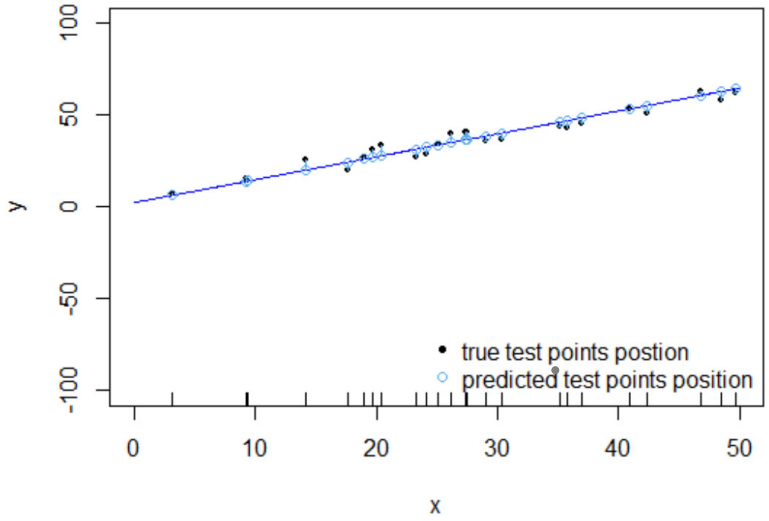
The rigid model ofteh “underfits” the data: the true underlying relationship is more complex.

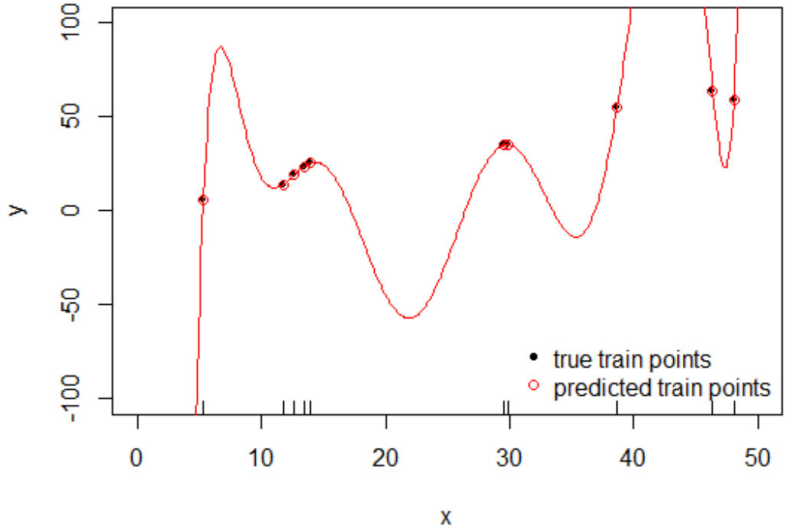
Get new train and test data and do all again

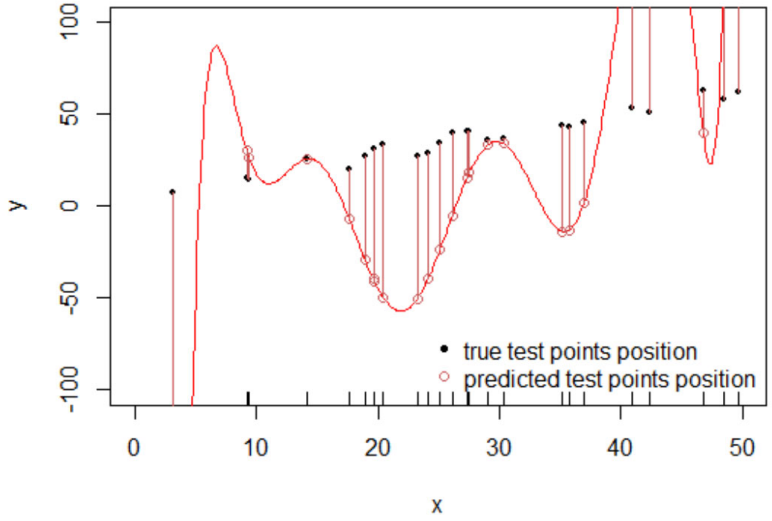
Check performance on **train** data



Check performance on **test** data



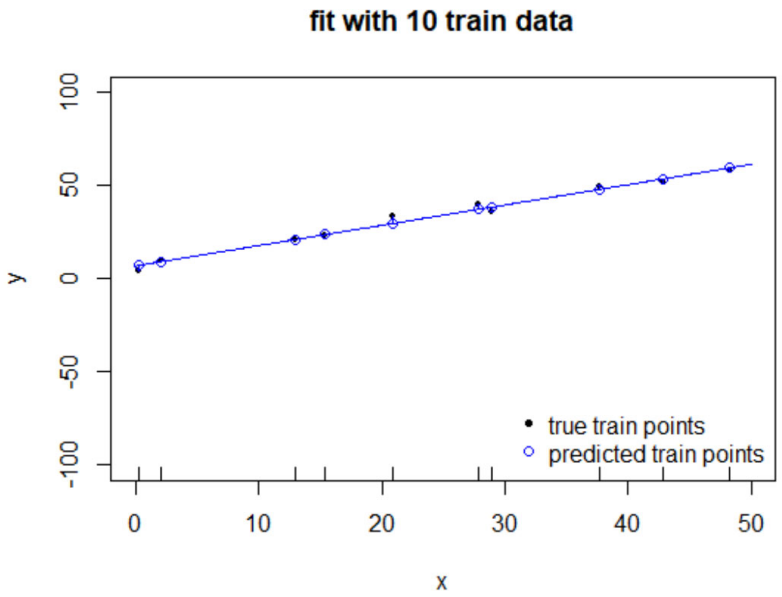




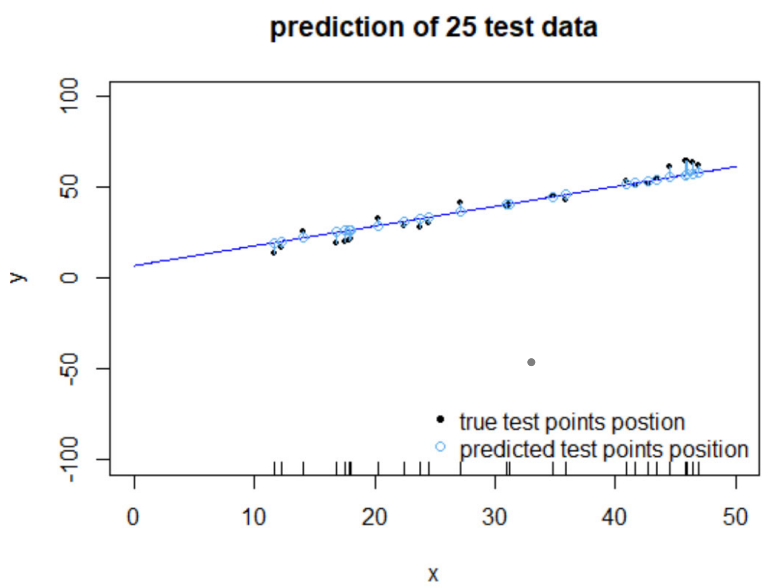
Note: The flexible model is on new train data very different to the run we saw before, the rigid model is very similar

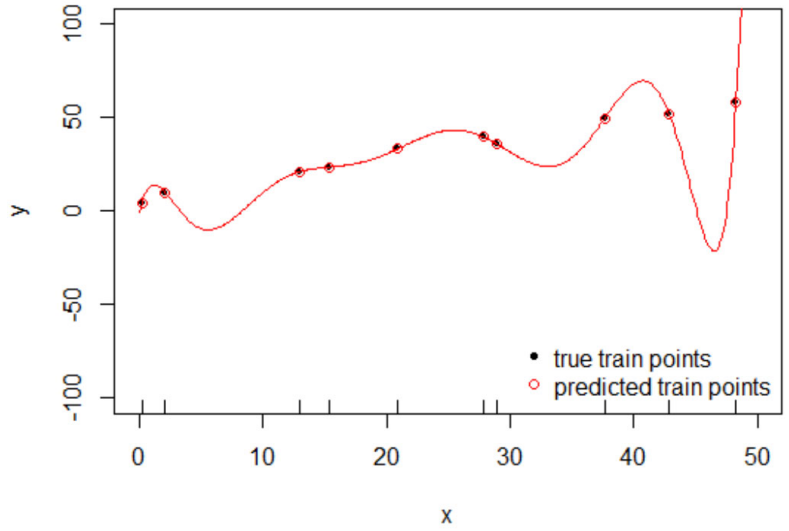
Get new train and test data and do all again

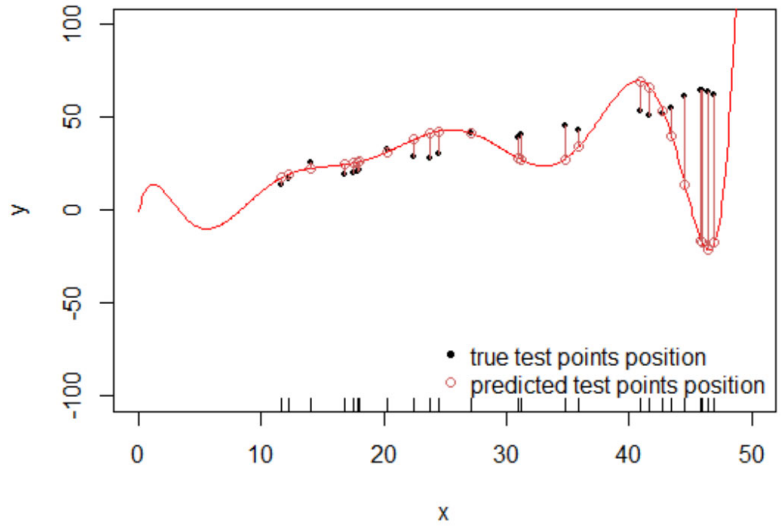
Check performance on **train** data



Check performance on **test** data







Note: The flexible model is on new train data very different to the run we saw before, the rigid model is very similar

Variance/Bias trade-off of flexible/rigid models

Our simulated data came from an ascending oscillating curve – see grey curve.

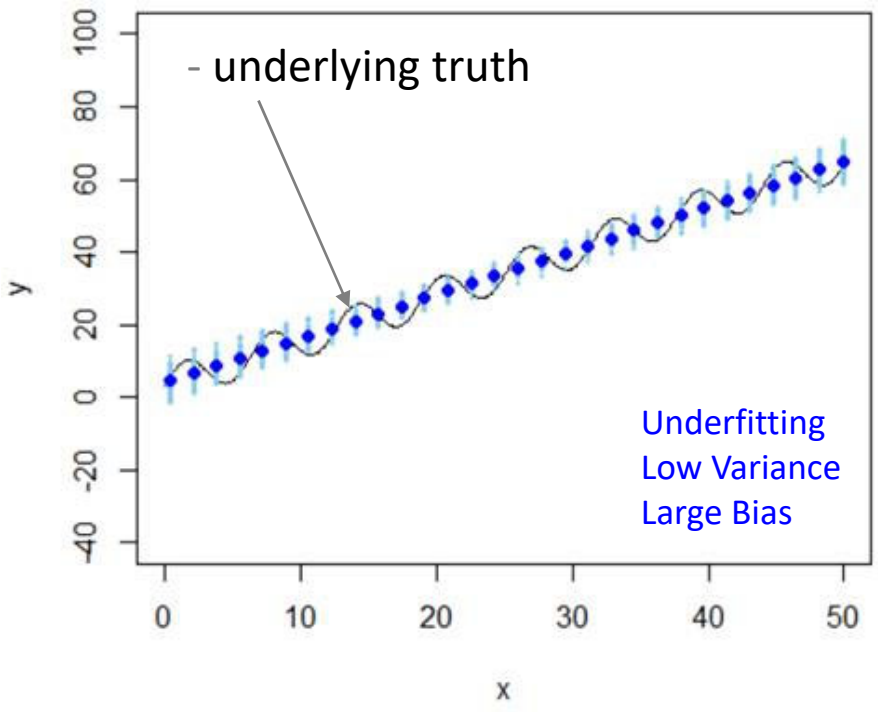
We sample 200 times a train set of 10 points and fit both models.

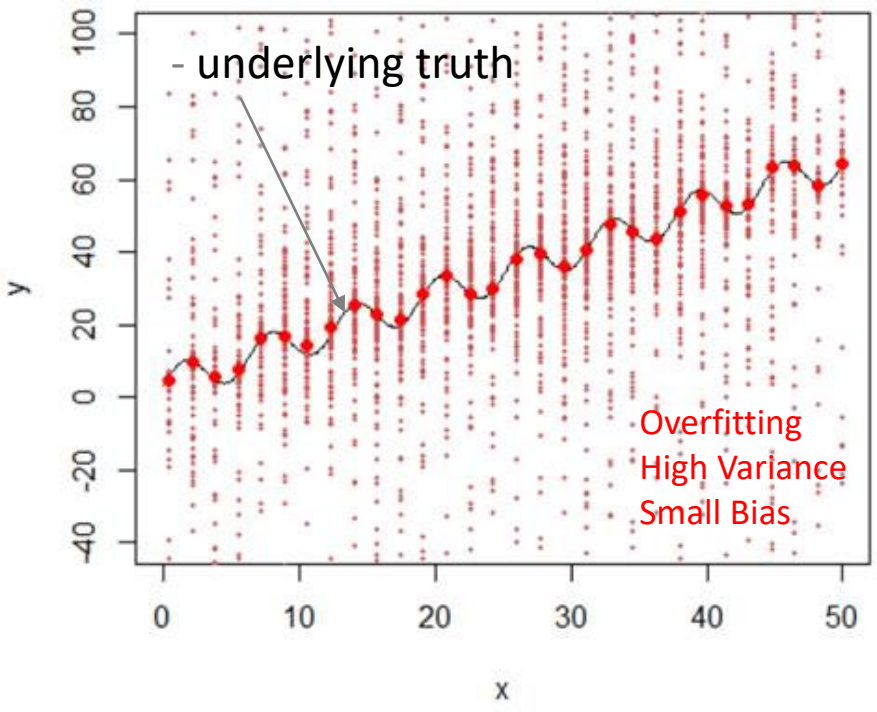
We evaluate both models in all 200 runs at 30 pre-selected x-positions – see tiny points.

We determine for both models at those x the mean of all 200 predictions – see fat points

→ Predictions of rigid model as much lower variance (see variation of tiny points)

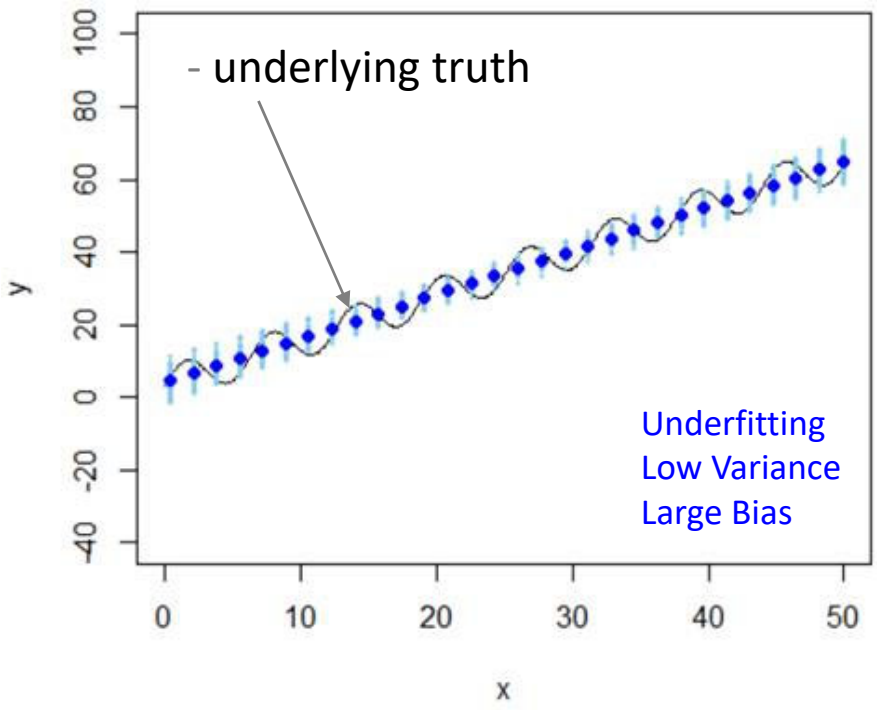
→ The mean predictions (fat points) of the rigid model has more systematic error = bias

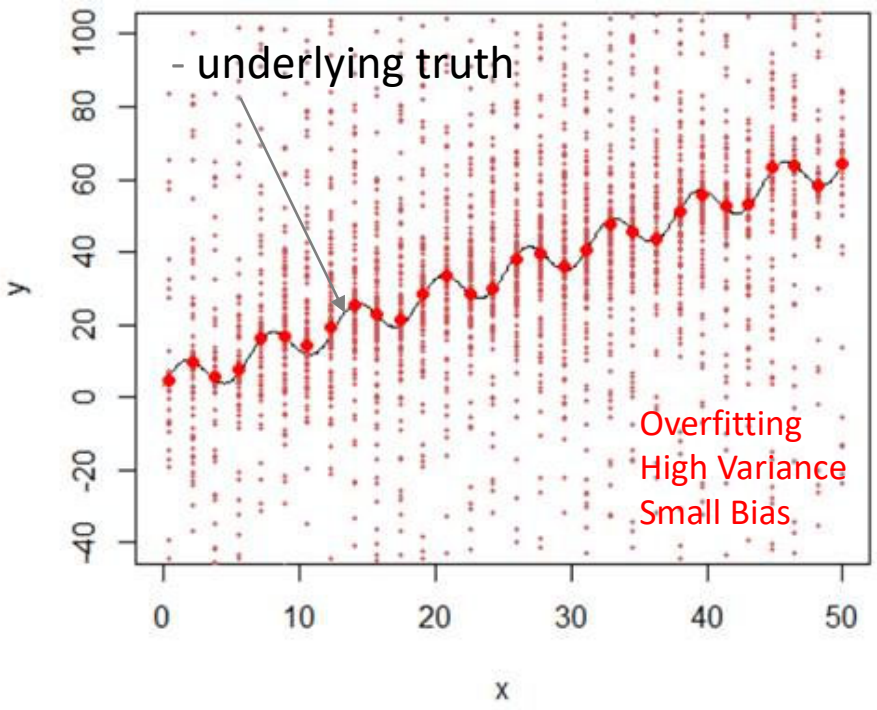




Which model gives better predictions?

We need to do a Variance/Bias trade-off





Here we probably would choose the rigid model, since here we can expect, that a (single) model, that we did fit to 10 train data points lead on test data predictions, that are quite close to the true value.

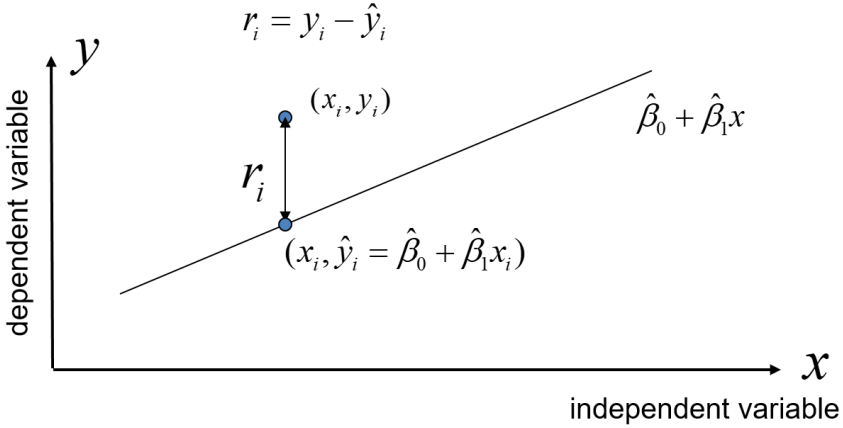
How to quantify the prediction performance of one specific model?

We use the **test error** and determine:

- the mean square error (MSE)
  or root mean square error (RMSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2$$

$$RMSE = \sqrt{MSE}$$



- the mean absolute error (MAE):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right|$$

- the mean absolute percentage error (MAPE)

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\frac{\left|y_i - \hat{y}_i\right|}{y_i}$$

A simple example in R: Splitting data in train and test

```
dat=cars  # this data set is part of standard R

head(dat)  # display head of data.frame
#>   speed dist
#> 1     4    2
#> 2     4   10
# ...

# Randomly split data into Training and Test data
set.seed(100)  # for reproducibility

# row indices for training data
trainingRowIndex <- sample(1:nrow(dat), 0.8*nrow(dat))

# train data set:
train <- dat[trainingRowIndex, ]

# test data set:
test <- dat[-trainingRowIndex, ]
```

Inspired by: http://r-statistics.co/Linear-Regression.html

A simple example in R: fitting a linear regression model

```
# Build the model on training data -
lmMod <- lm(dist ~ speed, data=train)  # build the model

# we get some performance measure from the summary output
summary(lmMod)
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -22.657      7.999  -2.833  0.00735 **
speed          4.316      0.487   8.863 8.73e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 15.84 on 38 degrees of freedom
Multiple R-squared:  0.674,     Adjusted R-squared:  0.6654
F-statistic: 78.56 on 1 and 38 DF,  p-value: 8.734e-11
```

Peformance metrics to evaluate the fit on the training data

Remark: Do residual analysis to check that the model assumptions are not violated.

Inspired by: http://r-statistics.co/Linear-Regression.html

A simple example in R: determine predictions on test data

```
# predict dist on test data
test$distPred <- predict(lmMod, newdata=test)

# test data with added column holding predictions
   speed dist  distPred
1      4    2 -5.392776
4      7   22  7.555787
8     10   26 20.504349
20    14   26 37.769100
26    15   54 42.085287
31    17   50 50.717663
37    19   46 59.350038
39    20   32 63.666225
40    20   48 63.666225
42    20   56 63.666225
```

Inspired by: http://r-statistics.co/Linear-Regression.html

A simple example in R: determine MSE, MAPE for test data

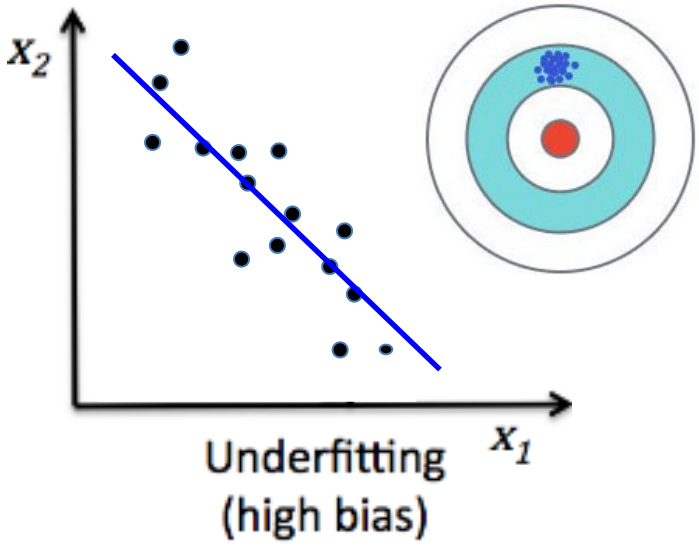
```
# predict dist on test data
test$distPred <- predict(lmMod, newdata=test)

# determine MSE and MAPE of test predictions:

MSE <- mean((test$dist - test$distPred)^2)

MAPE <- mean(abs((test$distPred - test$dist))/test$dist)
```

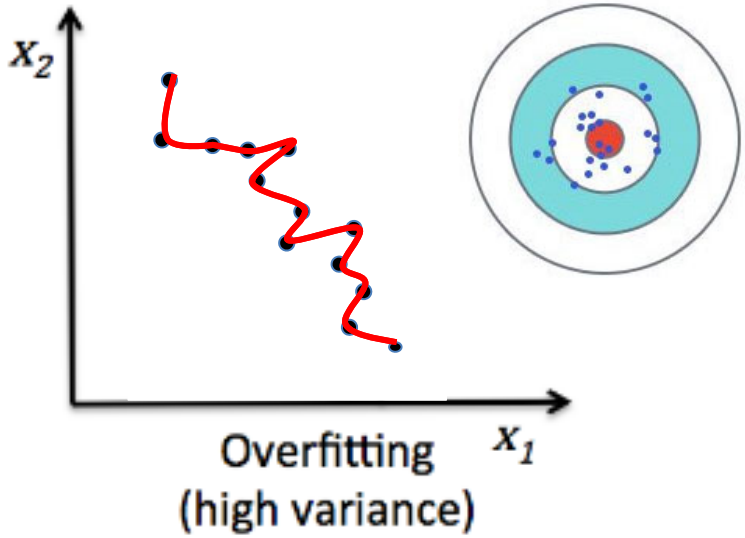
```
MSE
205.9653

MAPE
0.6995032
```

Performance metrics to evaluate the predictions **on the test data**

Inspired by: http://r-statistics.co/Linear-Regression.html

The concept of Bias and Variance of a regression model





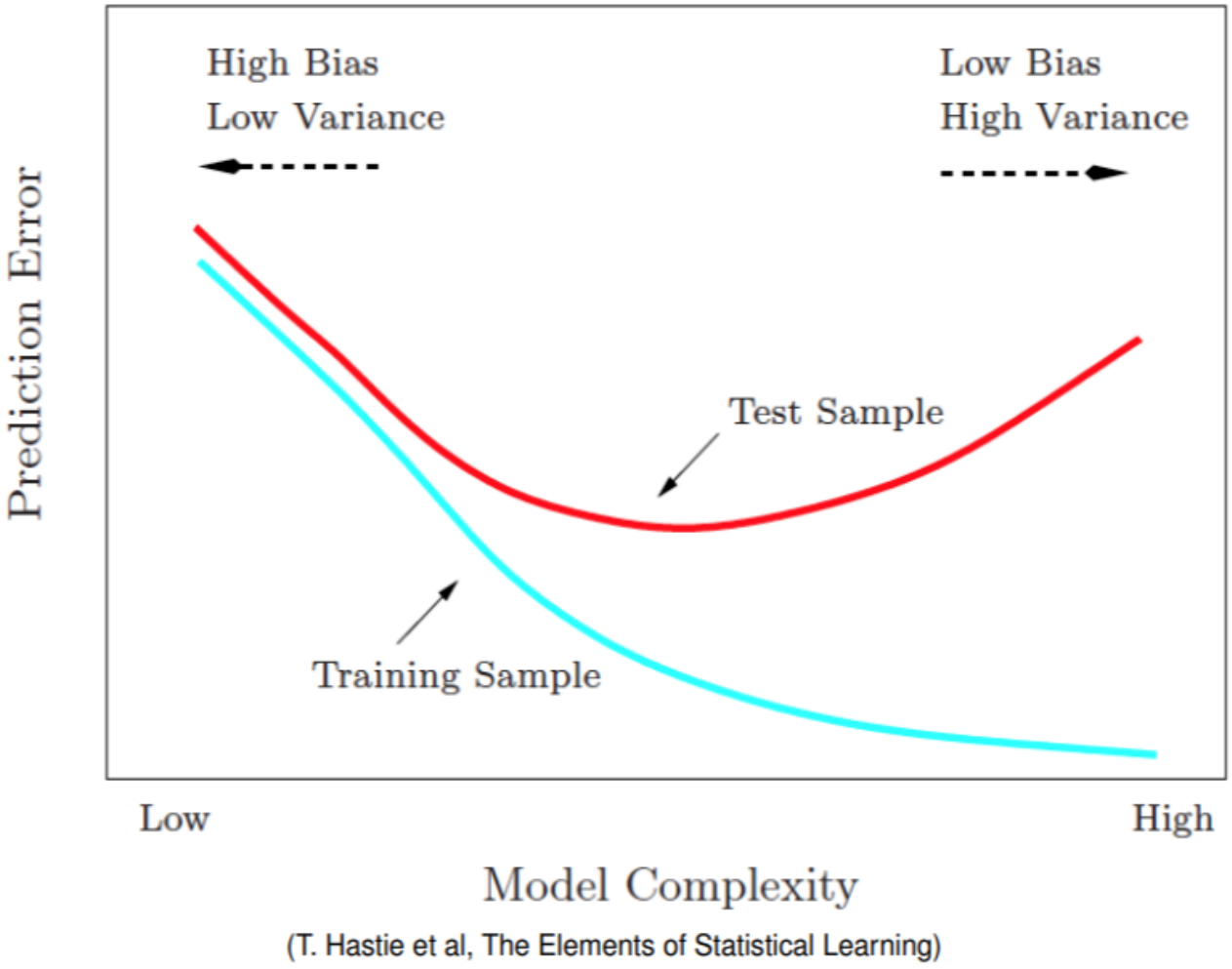
A under fitting model

- is not flexible enough for true data structure
- Some systematic error since the model assumes a too simple relationship (high bias)
- will not vary a lot if fitted to new train data (low variance)

A overfitting model

- is too flexible for data structure
- few errors on train set and non-systematic test errors (low bias)
- will vary a lot if fitted to new train data (high variance)

The concept of Bias and Variance tradeoff



(T. Hastie et al, The Elements of Statistical Learning)

See exercises for a more detailed discussion of this plot.

Let's simulate some data and split them in train and test set

```
n = 200 # number of observation
prop_train=0.25  # proportion of obs to use for train
p = 10  # number of predictors
error_sd = 2 # added noise
max_abs_coef = 2 # defines range of "slopes"
min_coef = 0.3  # we set "effect"=0 if slope<min_coef

# sample p coefficients between min and max
beta = matrix(runif(p, min=-max_abs_coef,
                       max=max_abs_coef), ncol=1)
# set beta to 0 if sampled beta was smaller 0.3
( beta = ifelse(abs(beta)<0.3, 0, beta) )

# data matrix:
x = matrix(rnorm(n*p), nrow=n, ncol=p)

# generate outcome y according to linear model plus error
y = x %*% beta + rnorm(n, sd=error_sd)

# built data frame holding predictors and outcome
dat_sim = data.frame(y, x)
head(dat_sim)
```

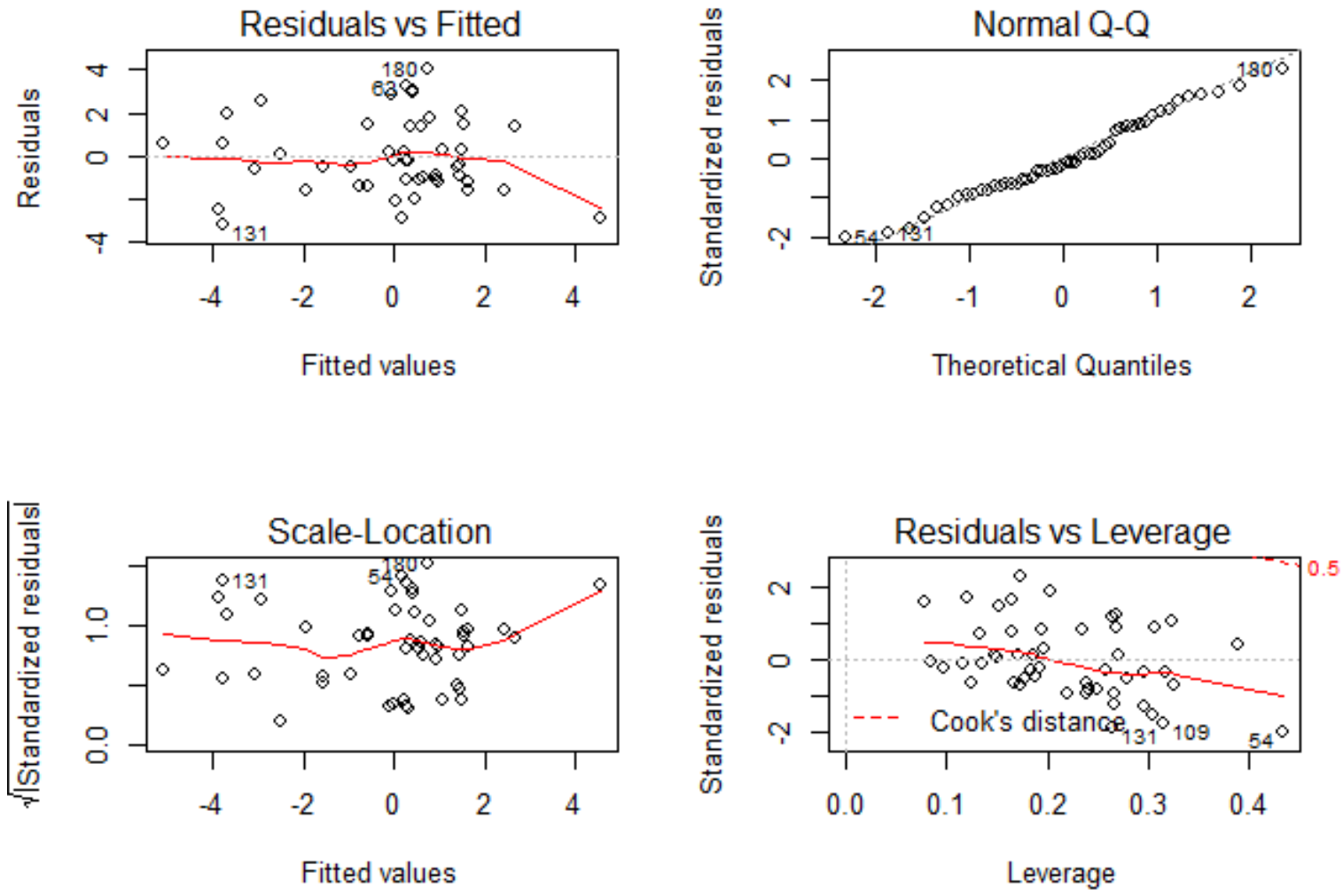
```
           y         X1          X2         X3         X4         X5         X6         X7         X8         X9        X10
 0.305239571  0.04266307 -1.81394212  1.1000002  0.68120622 -1.0160885 -1.3009742  1.5188241 -1.0362367  0.81158385  0.6602157
 0.762806963  0.01829321 -0.46685921  0.2346569 -2.66254636  0.1124149  1.8886849  1.3943780 -1.4605380 -0.19481145 -1.2991316
 0.006516694  0.97322521 -0.63294606  0.7620599 -0.47316238 -1.3423427 -0.9987358 -0.1026116  0.8962294 -0.33722534 -0.3553637
-6.344377597  2.00650835 -0.01336105  0.4250921  0.01623457  0.6719203 -0.9727473 -0.4315897  0.5360095 -0.19686322 -1.2474868
 4.633133787 -0.89475932 -0.03973259  1.2099400  0.73924955 -0.1493675 -0.9959355 -1.5872068 -0.4715230 -0.09685843  0.9034193
-1.085515170 -0.23540671 -0.12748278 -0.7399379  0.05949176 -0.8293857 -1.8514649 -1.0149369 -0.8859952 -0.64563866  1.6380714
...
```

```
# randomly split data in training and test
idx.tr = sample(1:nrow(dat_sim),
                as.integer(prop_train*nrow(dat_sim)))
train = dat_sim[idx.tr,]
test = dat_sim[-idx.tr,]
```

Let's fit a linear regression model for descriptive modelling

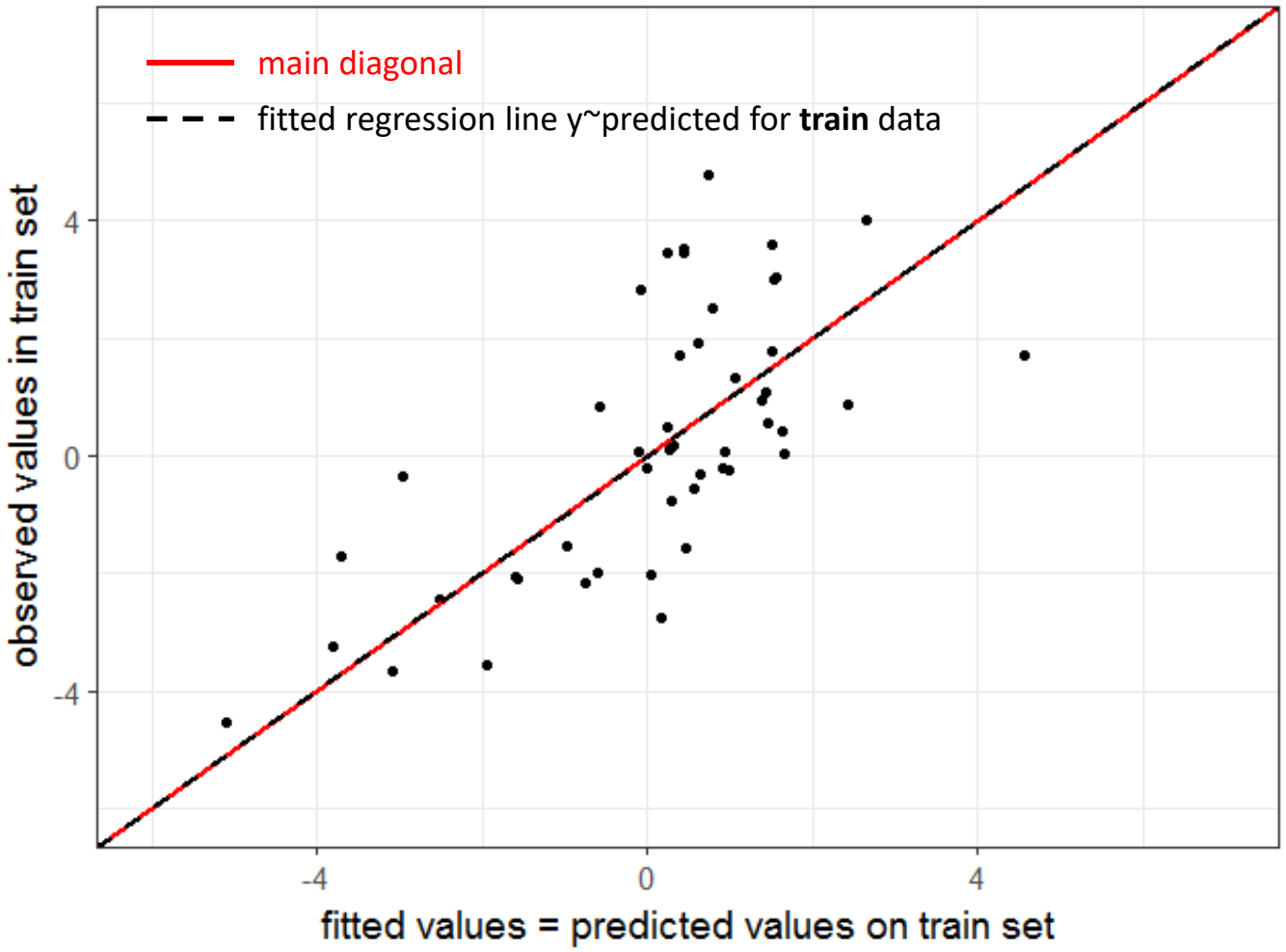
```
# now fit a regression model
fit_train = lm(y~. ,data=train )

# statistician's model check:
plot(fit_train)

# poor man's model check:
train$train_pred = predict(fit_train,
                           newdata=train)

# for unbiased fit we expect slope 1 for obs vs fitted
fit_train = lm(y ~ train_pred, data=train)

# check visually if fitted model is unbiased
ggplot(train , aes(x=train_pred, y=y)) +
  geom_point() +
  geom_abline(intercept=0, slope=1,
              linetype=1, colour='red', size=1) +
  geom_abline(intercept=coef(fit_train)[1],
              slope=coef(fit_train)[2],
              linetype=2, colour='black', size=1) +
  theme_bw(base_size = 14) +
  xlim(my_ymin,my_ymax) +
  ylim(my_ymin,my_ymax) +
  ylab("observed values in train set") +
  xlab("fitted values = predicted values on train set")
```

Statisticians descriptive model check: residual analysis



Residual plots look o.k., what is expected since true model is fitted to simulated data.

Poor man's descriptive model check: observed vs fitted

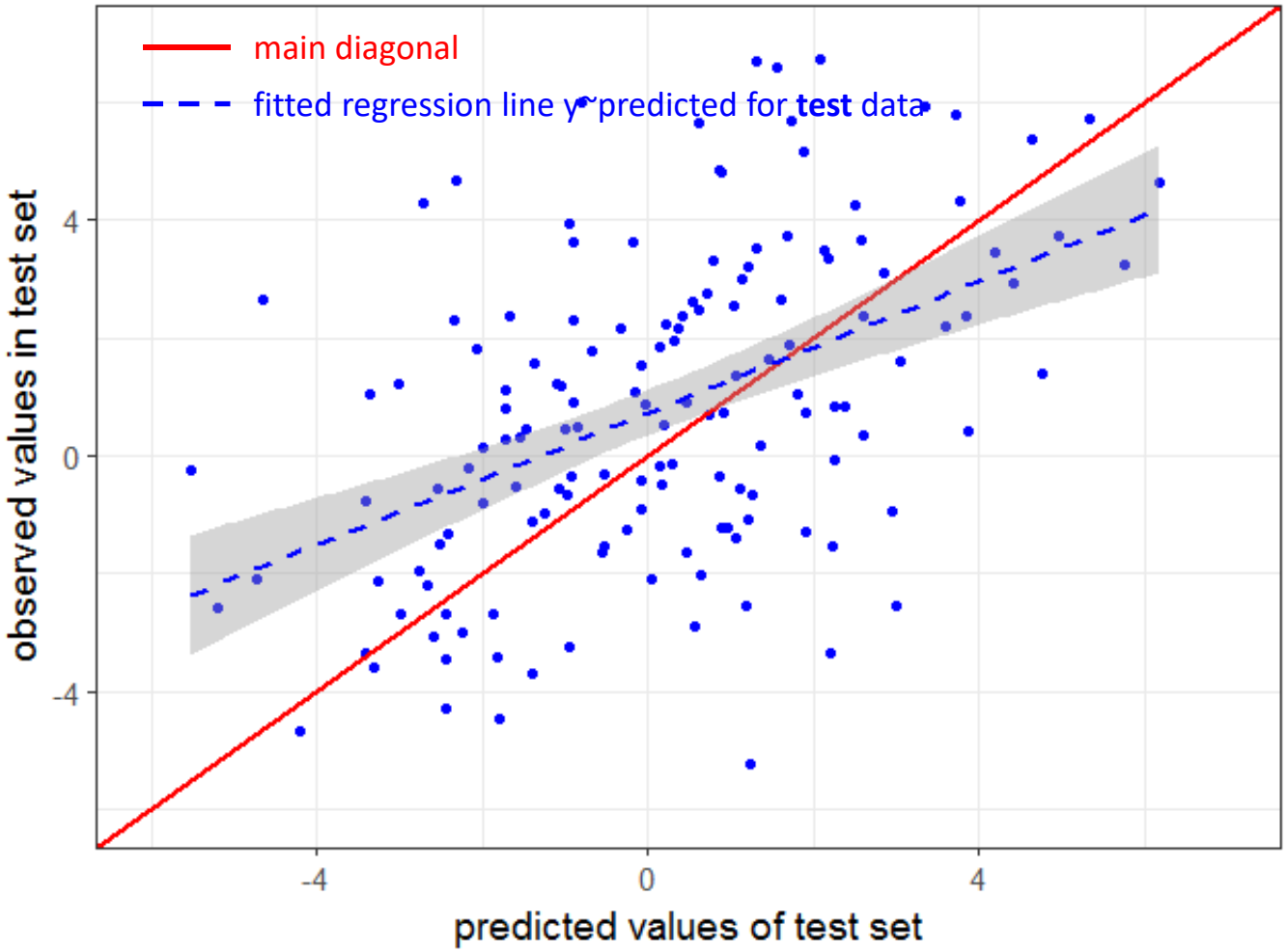


Slope of fitted line for observed vs fitted is 1 proving that linear regressions produces unbiased fitted values.

Let's use the regression model as predictive model

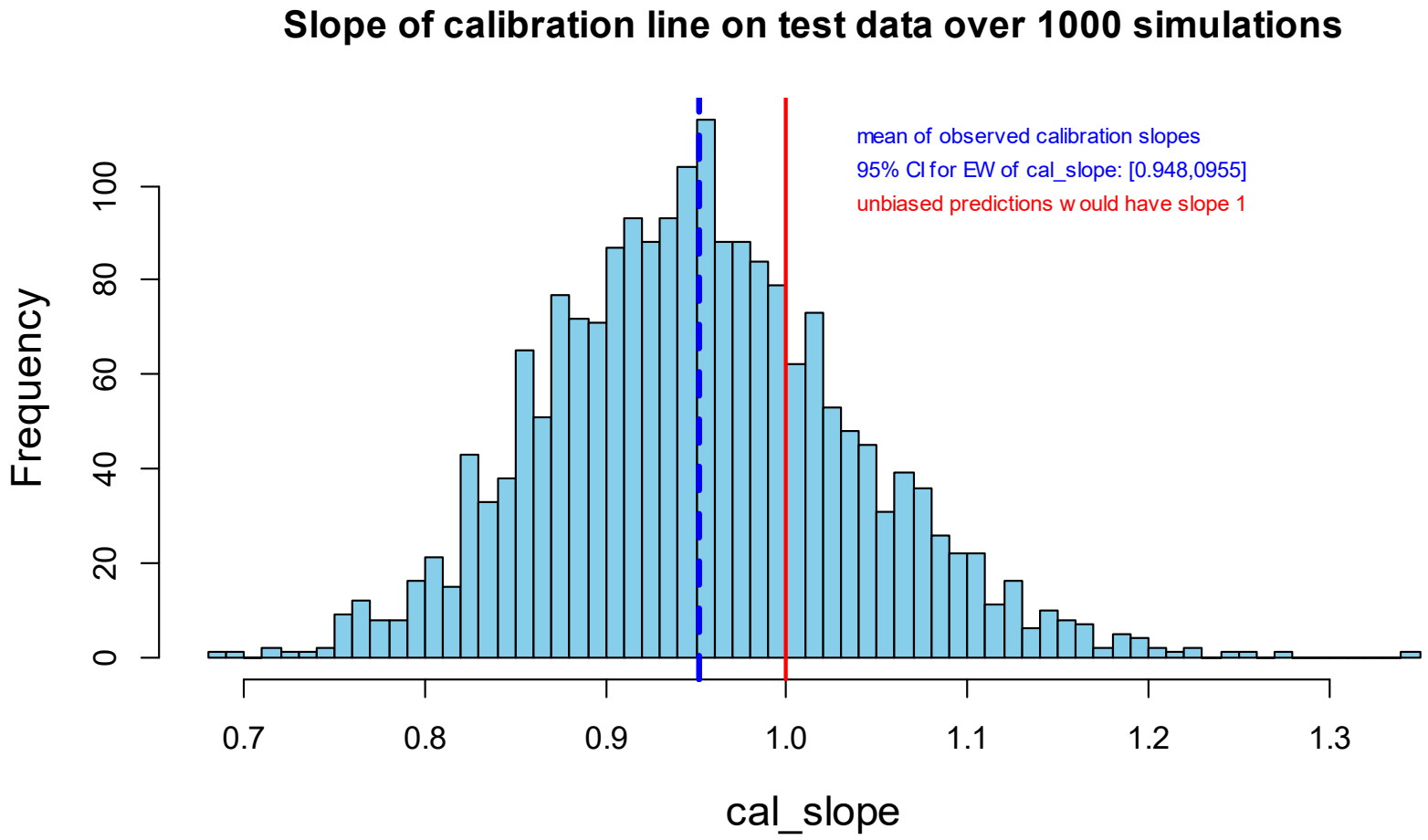
```
# fit a regression model on train data
fit_train = lm(y~. ,data=train )

## predict new test data with fit_train
test$test_pred = predict(fit_train, newdata=test)

# check if predictions on new data are unbiased
ggplot(test , aes(x=test_pred, y=y)) +
  geom_point(colour='blue') +
  geom_abline(intercept=0, slope=1,
              linetype=1, colour='red', size=1) +
  geom_smooth(method="lm",linetype=2, colour='blue' ) +
  theme_bw(base_size = 14) +
  xlim(my_ymin,my_ymax) +
  ylim(my_ymin,my_ymax) +
  ylab("observed values in test set") +
  xlab("predicted values of test set")
```

Let's check if predictions are unbiased: calibration plot



Slope of fitted line for observed vs predicted values is smaller that 1 meaning that linear regressions produces to extreme predicted values.

Are predictions always to extreme?



The simulation shows that on average linear regressions produces to extreme predicted values. But we have sample variation and the bias varies from run to run and sometimes the predictions are even not extreme enough.

In which cases are "naïve" predictions biased on new data?

If the regression fit (on training data) is not very good (small F statistics) indicating that the model rather overfits the data as it easily happens for

- lot of noise – large residual error
- small (training) data set
- small effects – small coefficients
- many predictors

→ see R simulation

Observations in R simulation:

- For p>=2 predictors and small effects we can observe this bias in the predictions (mean calibration slope b is significantly smaller than 1 when using the a model fitted on train to predict test data)
- For p=1 predictor (simple regression) we cannot estimate the expected value of the calibration slope b to show b <1
- The distribution of the simulated calibration slopes is not Normal but is rather compatible with a t-distribution with df=p

Houwelingen2001 p.25: "It might be expected that shrinkage in the regression setting only works if there are at least three covariates yielding four unknown parameters."

Shrinkage leads to calibrated predictions on new data

- Naïve Least Square Prediction uses the model fitted on train data also to predict new data:

$$\hat{\mathbf{y}}_{LS} = \hat{\boldsymbol{\alpha}}_{LS} + \mathbf{x} \cdot \hat{\boldsymbol{\beta}}_{LS}$$

- Since we observe that predictions on new data are on average to extreme, we need to shrink them towards the mean to get calibrated predictions

- This can be achieved by shrinking the coefficients in the regression model (w/o loss of generality we assume that all predictors are centered).

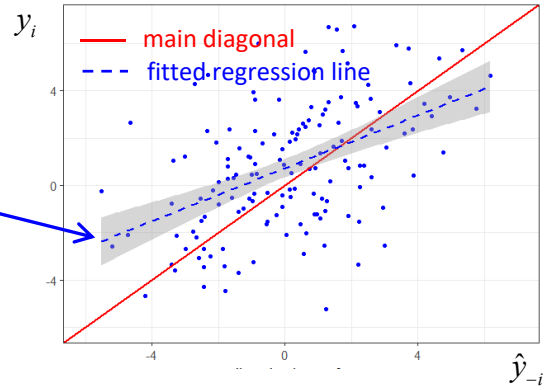
- Recalibrated prediction achieved by shrinking the coefficients with a shrinkage factor $c \in [0,1]$:

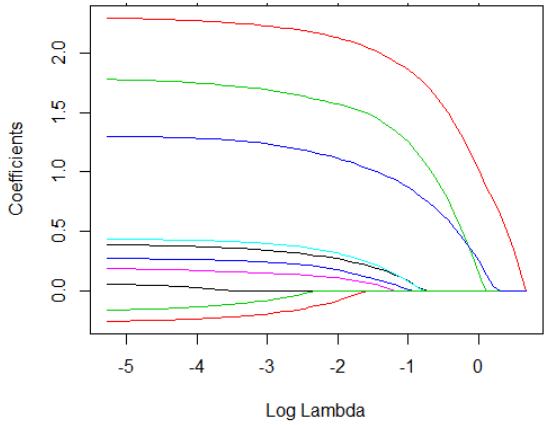
$$\hat{\mathbf{y}}_{\text{recalibrated}} = \hat{\boldsymbol{\alpha}}_{LS} + \mathbf{X} \cdot \left(c \cdot \hat{\boldsymbol{\beta}}_{LS} \right)$$

How to determine how much we need to shrink the coefficients?

- Based on the global F-statistics: $\hat{c} = \max\left(1 - \frac{1}{F}, 0\right)$
  (see Copas 1997)

- Or better through cross validation:
  - regress y_i on $\hat{y}_{-i}$ where $\hat{y}_{-i}$ is the naive LS prediction where the LS model was fitted while omitting the i-th observation: the slope of this regression is the estimated shrinkage factor c.
  - Use penalized regressions such as LASSO or ridge regression resulting in shrinked coefficients by applying penalty on large coefficients, where the amount of penalty is tuned by a parameter λ that is optimized via cross validation





Ridge regression

- Recall that the least squares fitting procedure estimates $\beta_0, \beta_1, \ldots, \beta_p$ using the values that minimize

$$\text{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 .$$

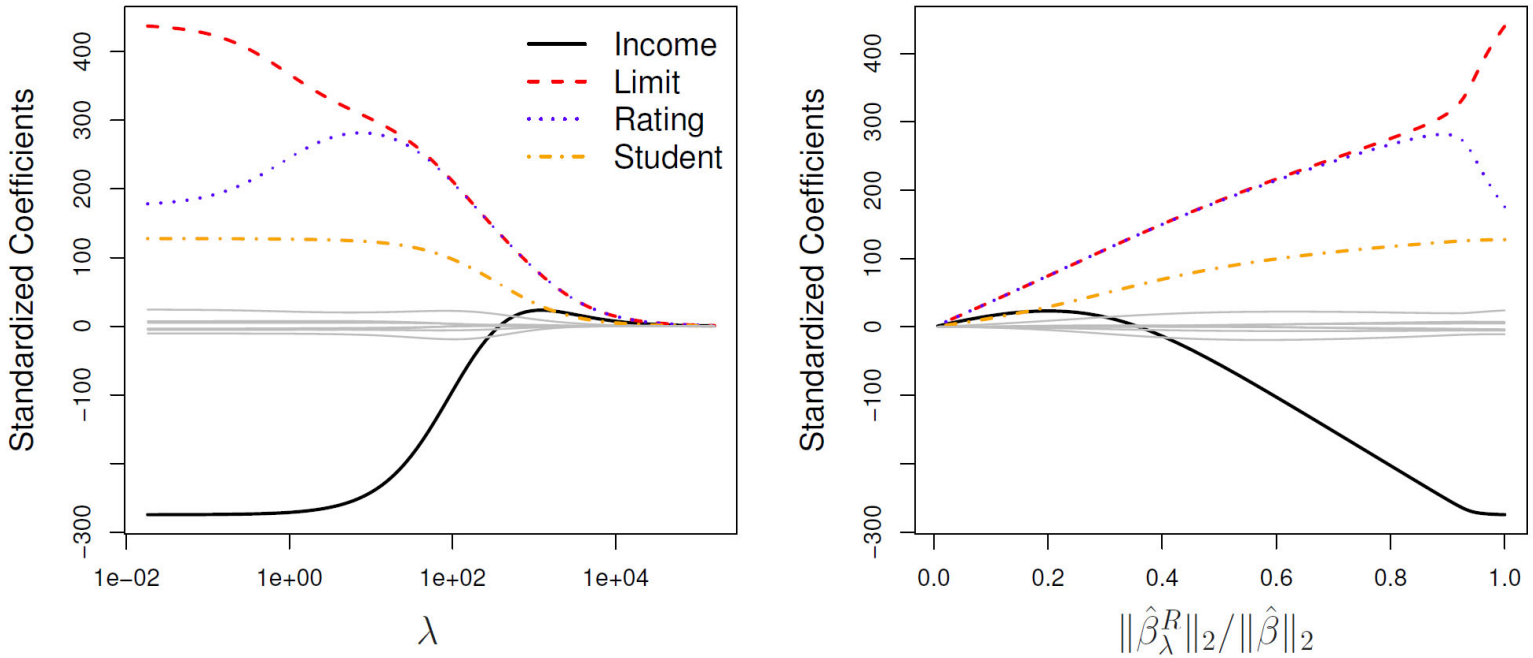
- In contrast, the ridge regression coefficient estimates $\hat{\beta}^R$ are the values that minimize

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = \text{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2,$$

where $\lambda \geq 0$ is a *tuning parameter*, to be determined separately.

ISLR chapter 6 slides Hastie Tibshirani

Coefficient paths when tuning penalty in Ridge Regression



ISLR chapter 6 slides Hastie Tibshirani

Going from ridge regression to Lasso

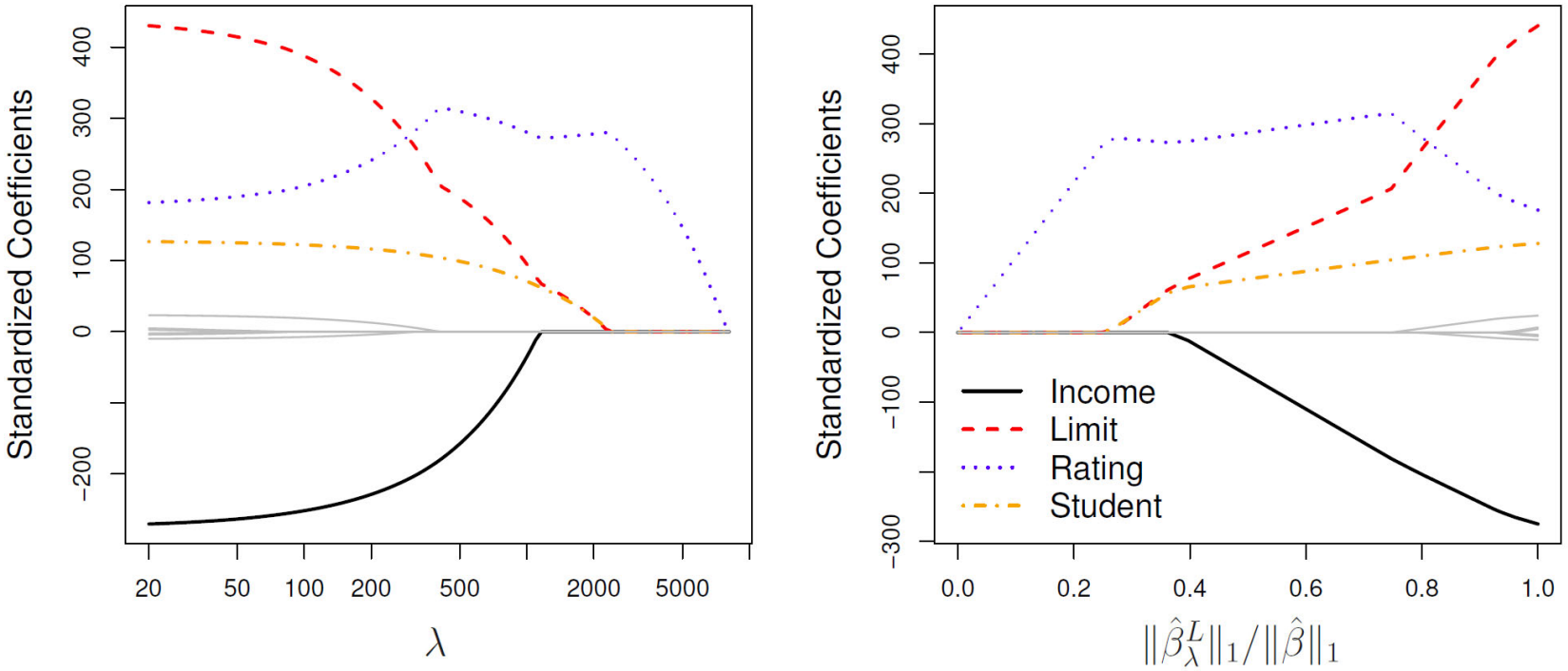
- Ridge regression does have one obvious disadvantage: unlike subset selection, which will generally select models that involve just a subset of the variables, ridge regression will include all p predictors in the final model
- The *Lasso* is a relatively recent alternative to ridge regression that overcomes this disadvantage. The lasso coefficients, $\hat{\beta}_\lambda^L$, minimize the quantity

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| = \text{RSS} + \lambda\sum_{j=1}^{p}|\beta_j|.$$

- In statistical parlance, the lasso uses an ℓ_1 (pronounced "ell 1") penalty instead of an ℓ_2 penalty. The ℓ_1 norm of a coefficient vector β is given by $\|\beta\|_1 = \sum|\beta_j|$.

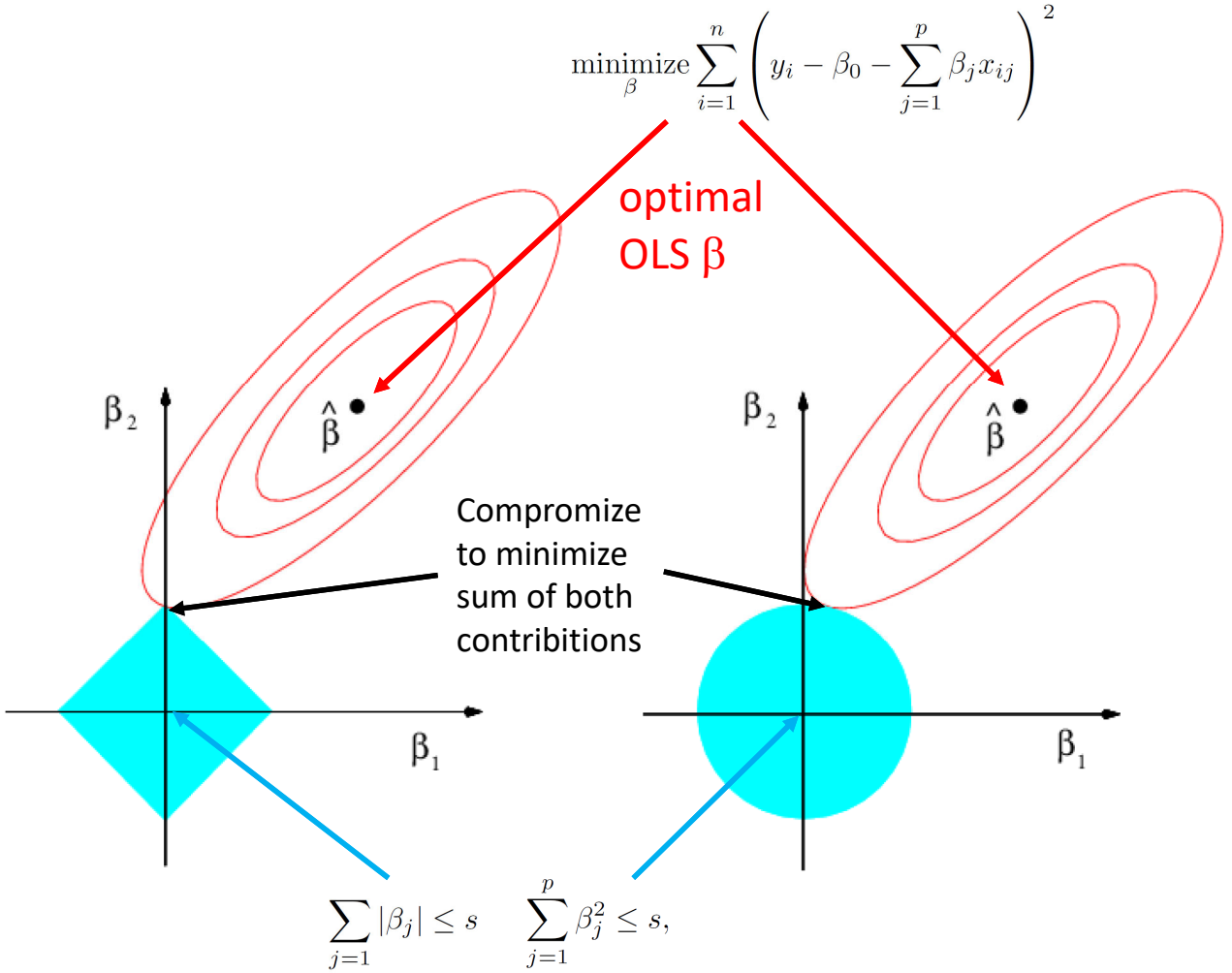
ISLR chapter 6 slides Hastie Tibshirani

Coefficient paths when tuning penalty in Ridge Regression



ISLR chapter 6 slides Hastie Tibshirani

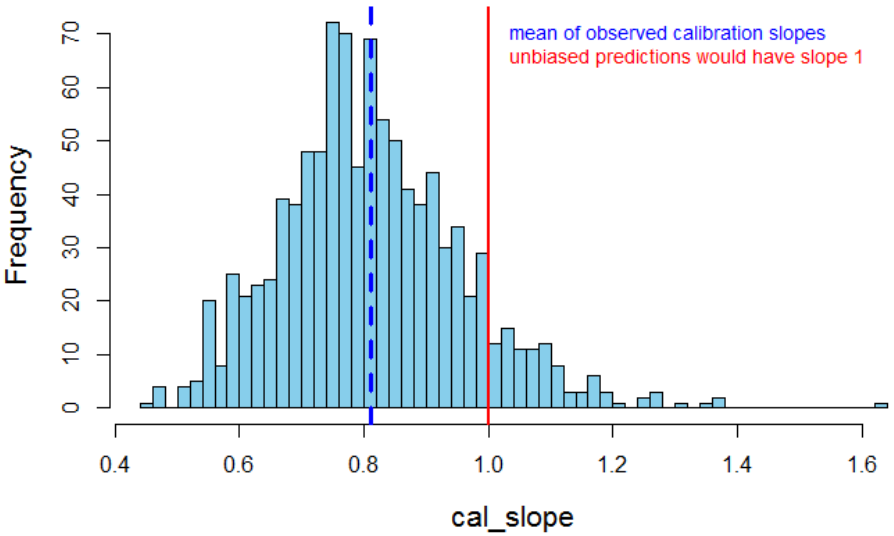
Geometric interpretation of objective in ridge and lasso



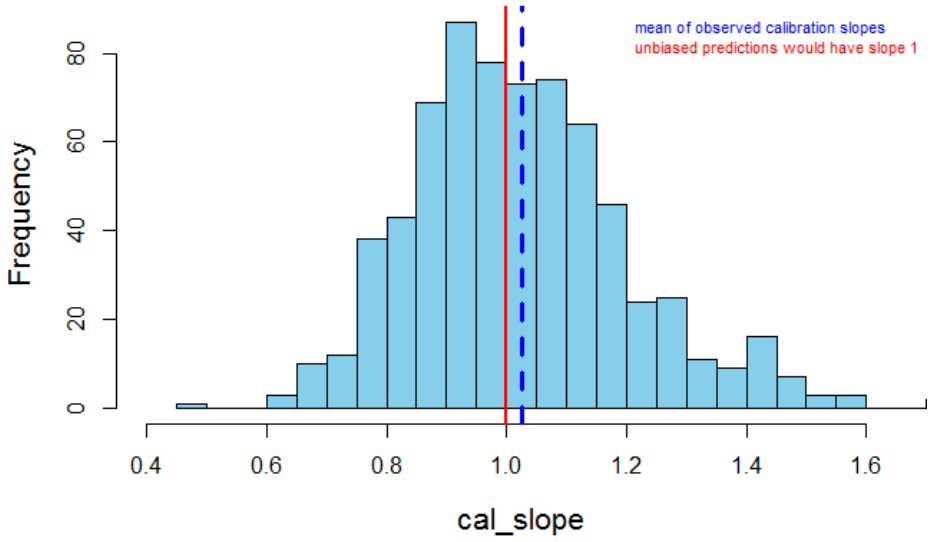
The point of compromise, where the sum of both loss components is minimal is for lasso often on an axis where one coefficient is zero – for ridge this is not likely to be the case.

Compare calibration of LS-lm versus cv-lasso model

(unshrinked) LS **lm** model



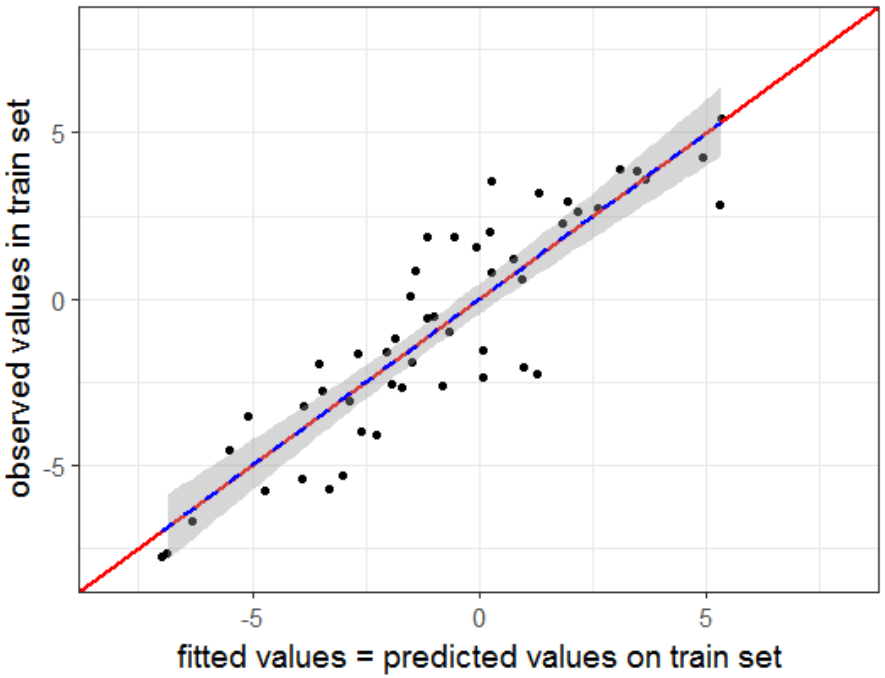
cv-shrinked **lasso** model

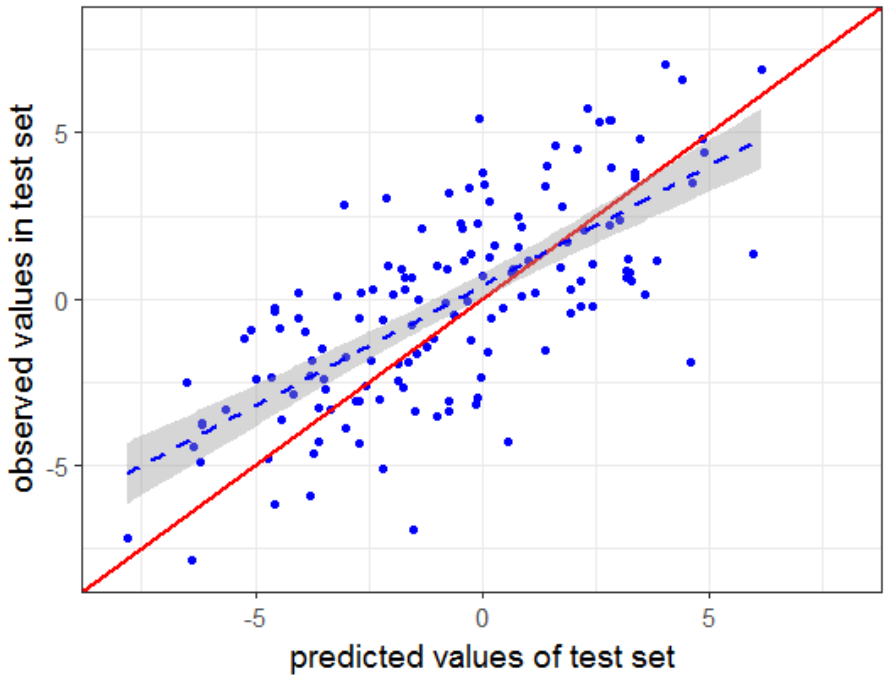


The linear model from the simulation (10 predictors) is better calibrated after Lasso shrinkage.

```
library(glmnet)
fit_lasso = glmnet(x=X, y=Y, alpha=1)
plot(fit_lasso, xvar = "lambda")
crossval =  cv.glmnet(x=X, y=Y)
plot(crossval)
opt_la = crossval$lambda.min
fit1 = glmnet(x=X, y=Y, alpha=1,
                    lambda=opt_la )
```

Calibration slope for train and test predictions





Calibration fit: $y = a + b \cdot \hat{y}$ where b is the calibration slope

Using the model that was fitted on training set for prediction we get:

always a calibration-slope = 1

when predicting training data

on average a calibration-slope < 1

when predicting test data

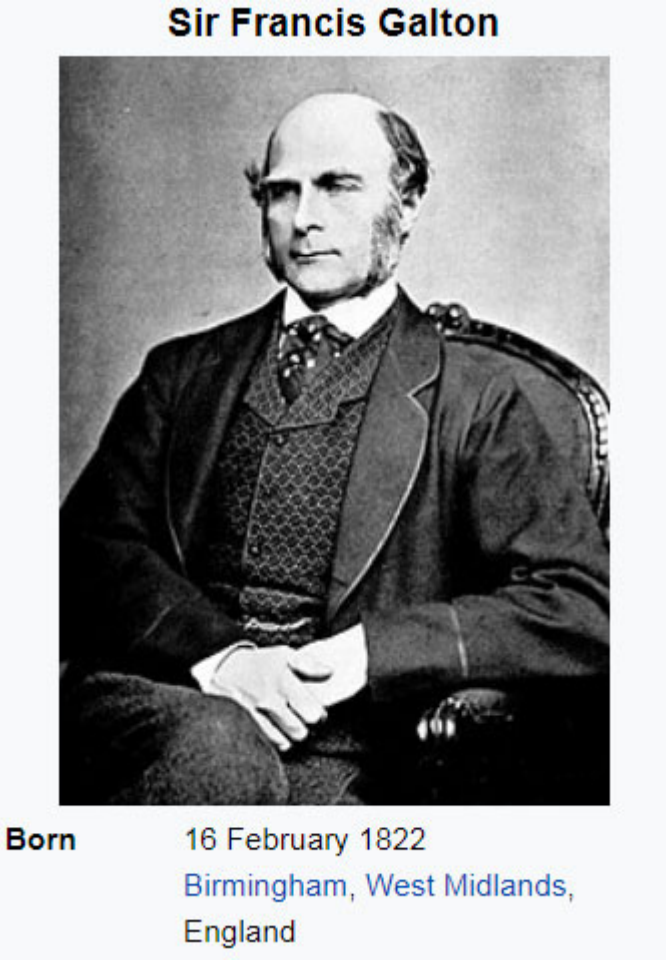
Regression to the mean

Where does the term "regression" come from?

The concept of regression comes from genetics and was popularized by Sir Francis Galton during the late 19th century with the publication of *Regression towards mediocrity in hereditary stature*.

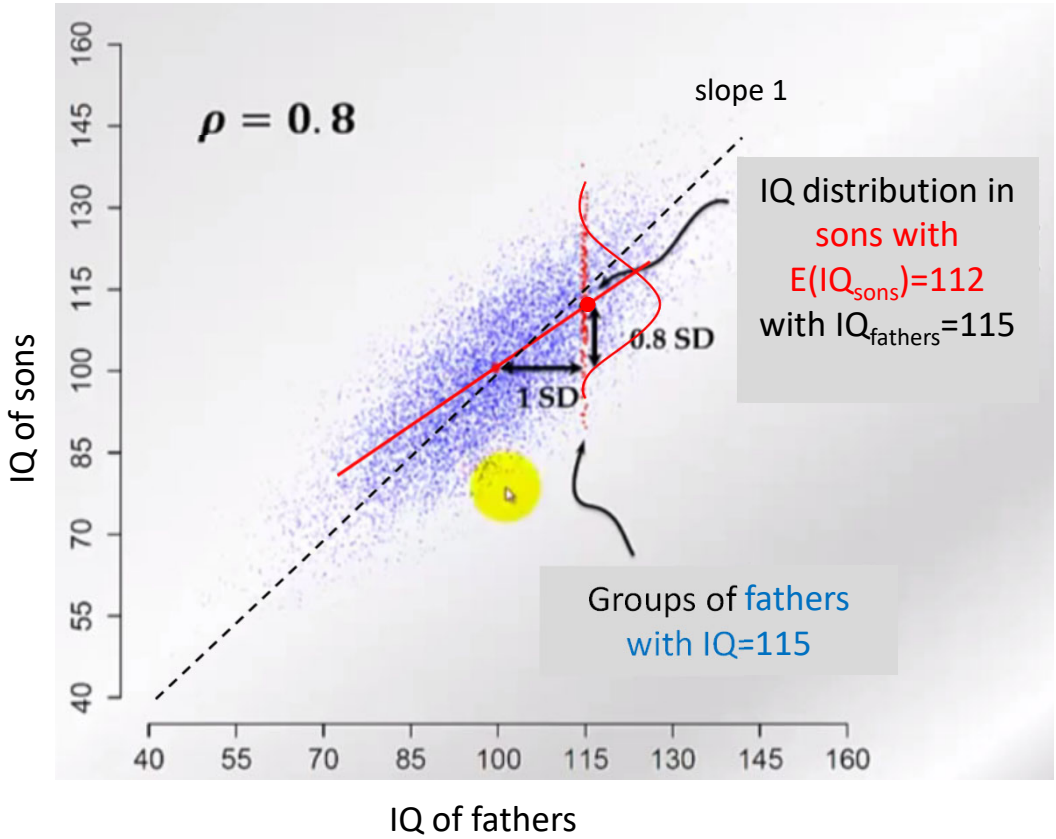
Galton observed that extreme characteristics (e.g., height) in parents are not passed on completely to their offspring. Rather, the characteristics in the offspring *regress* towards a *mediocre* point (a point which has since been identified as the mean).

Source: https://en.wikipedia.org/wiki/Francis_Galton



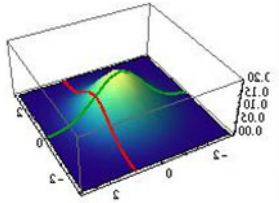
Galton's discovery of the regression line

Remark: Correlation of IQs of parents and children is only 0.42 https://en.wikipedia.org/wiki/Heritability_of_IQ



$$X1 \sim N\left(\mu_1 = 100,\ \sigma_1^2 = 15^2\right)$$
$$X2 \sim N\left(\mu_1 = 100,\ \sigma_1^2 = 15^2\right)$$

$$\begin{pmatrix} X1 \\ X2 \end{pmatrix} \sim N\left(\begin{pmatrix} 100 \\ 100 \end{pmatrix}, \begin{pmatrix} 15^2 & \mathrm{cov}(X1,X2) \\ \mathrm{cov}(X1,X2) & 15^2 \end{pmatrix}\right)$$

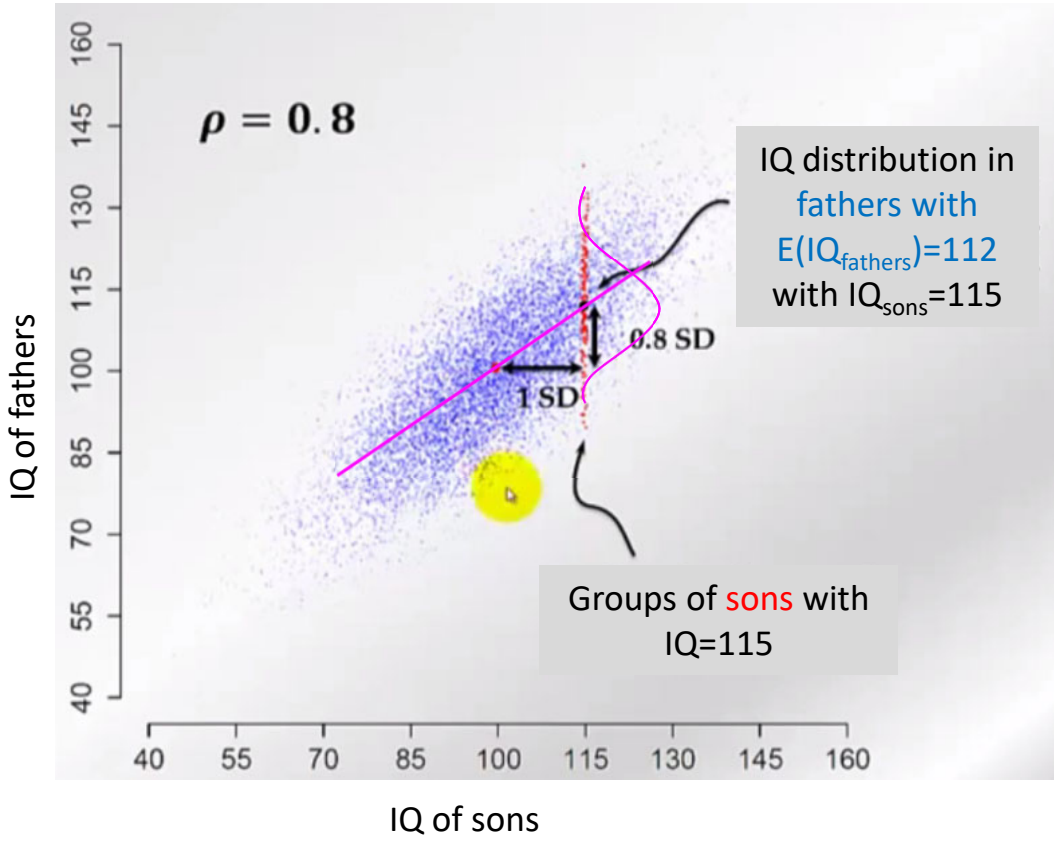


For each group of father with fixed IQ, the mean IQ of their sons is closer to the overall mean IQ (100) -> Galton aimed for a causal explanation.

All these predicted $E(IQ_{son})$ fall on a "regression line" with slope<1.

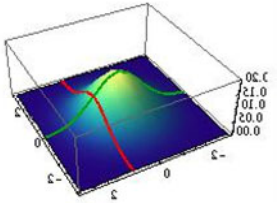
Image credits (changed): https://www.youtube.com/watch?v=aLv5cerjV0c

Galton's discovery of the regression to the mean phenomena



$$X1 \sim N\left(\mu_1 = 100,\ \sigma_1^2 = 15^2\right)$$
$$X2 \sim N\left(\mu_1 = 100,\ \sigma_1^2 = 15^2\right)$$

$$\begin{pmatrix} X1 \\ X2 \end{pmatrix} \sim N\left(\begin{pmatrix} 100 \\ 100 \end{pmatrix}, \begin{pmatrix} 15^2 & \text{cov}(X1,X2) \\ \text{cov}(X1,X2) & 15^2 \end{pmatrix}\right)$$

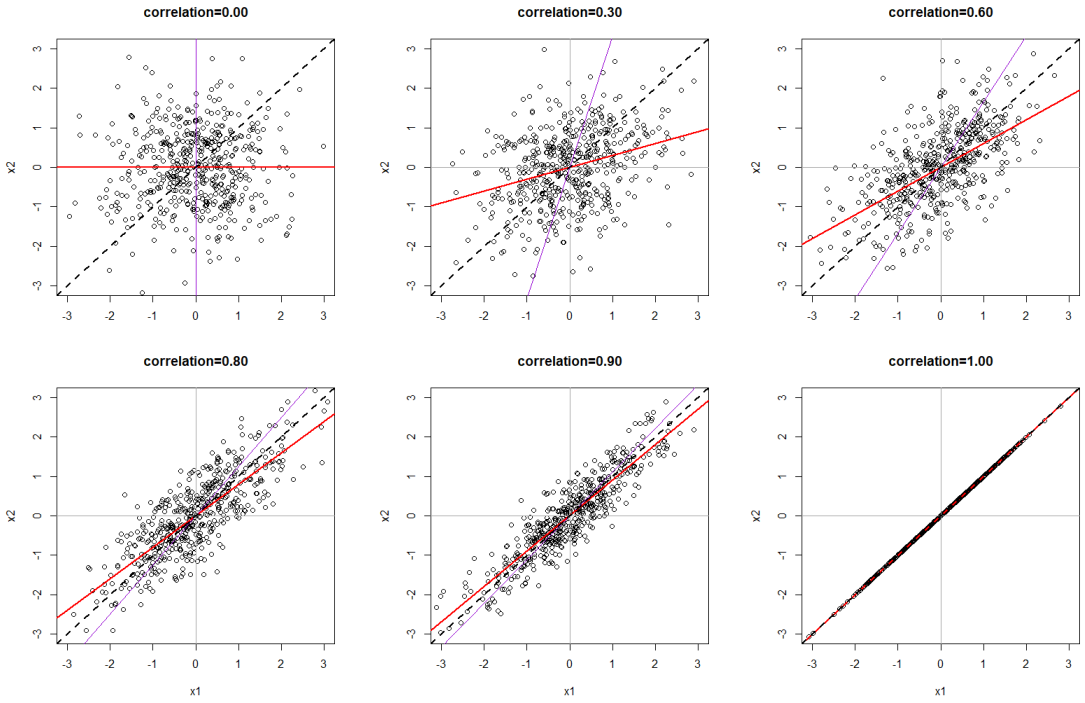


After switching the role of sons's IQ and father's IQ, we again see that $E(IQ_{fathers})$ fall on the regression line with the same slope <1.

There is no causality in this plot -> causal thinking seemed unreasonable.

Image credits (changed): https://www.youtube.com/watch?v=aLv5cerjV0c

Pearson's mathematical definition of correlation unmasks "regression to the mean" as statistical phenomena



The correlation c of a bivariate Normal distributed pair of random variables are given by the slope of the regression line after standardization!

c quantifies strength of linear relationship and is only 1 in case of deterministic relationship.

After standardization of the RV:

$$X1 \sim N\left(\mu_1 = 0,\ \sigma_1^2 = 1^2\right)$$
$$X2 \sim N\left(\mu_2 = 0,\ \sigma_2^2 = 1^2\right)$$

$$\begin{pmatrix} X1 \\ X2 \end{pmatrix} \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \sigma_{X1}^2 = 1 & c \\ c & \sigma_{X2}^2 = 1 \end{pmatrix}\right)$$

Regression line equation:

$$\hat{X}_2 = E\left(X_2 \mid X_1\right) = \beta_0 + \beta_1 \cdot X_1$$

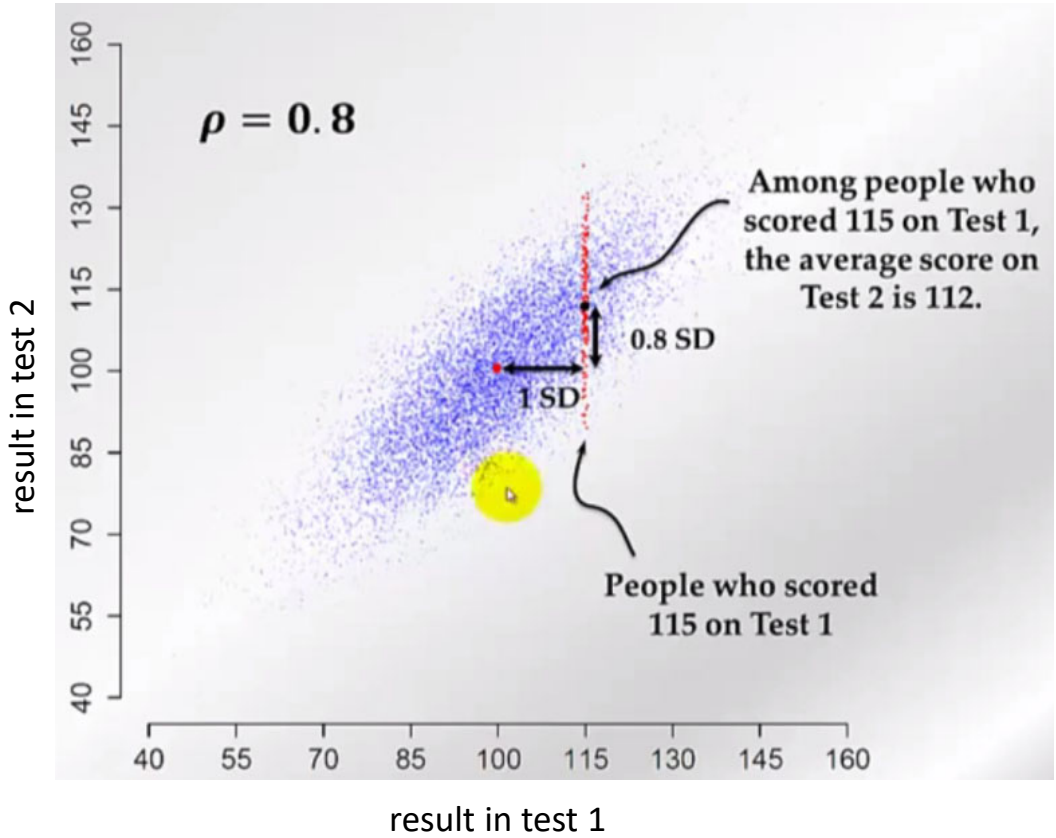
$$\beta_1 = c \cdot \frac{\sigma_2}{\sigma_1} \overset{\text{stand.}}{=} c$$

β_1 quantifies regression to the mean

$$\beta_0 = \mu_2 - \beta_1 \cdot \mu_1 \overset{\text{stand.}}{=} 0$$

$$c = \frac{\frac{1}{n-1}\sum_{i=1}^{n}(x_{i1} - \bar{x}_1)\cdot(x_{i2} - \bar{x}_2)}{\mathrm{sd}(x_1)\cdot\mathrm{sd}(x_2)}$$

Regression to the mean occurs in all test-retest situations



Retesting a extreme group (w/o intervention in between) in a second test leads in average to a results that are closer to the overall-mean -> to assess experimentally the effect of an intervention also a control group is needed!

Summary

- Linear regression models can be used for descriptive, explanatory or predictive modeling – the purpose determines the best fit strategy
- To develop a predictive model we split our data into train, validation and test set to avoid overfitting (or do cross validation)
- Choosing between flexible and rigid models is based on variance/bias tradeoff which trades between over/under-fitting
- Prediction performance can be quantified by the MSE on new data
- Least square parameter in linear regression models are often not optimal to achieve best prediction performance but need to be shrinked which can easily be done via ridge regression or lasso.
- Regression to the mean is a statistical phenomena resulting in less extreme values in second test of a test-retest situations and is a reason for why a control group is needed