Methods 3: Multilevel Statistical Modeling and Machine Learning

Week 05: *Explanation and Prediction*
October 2, 2024

The course plan

Week 1: *Introduction*

Instructor sessions: *Setting up* R *and* Python *and recollection of the general linear model*

Week 2: *Multilevel linear regression*

Instructor sessions: *Modelling subject level effects – and how do they differ from group level effects?*

Week 3: *Link functions and fitting generalised linear multilevel models*

Instructor sessions: *What to do when the response variable is not continuous?*

Week 4: *Evaluating Generalised linear mixed models*

Instructor sessions: *How do we assess how models compare to one another?*

Week 5: *Explanation and Prediction*

Instructor sessions: *Code review*

Week 6: *Mid-way evaluation and Machine Learning Intro*

Instructor sessions: *Getting Python Running*

Week 7: *Linear regression revisited (machine learning)*

Instructor sessions: *How to constrain our models to make them more predictive*

Week 8: *Logistic regression revisited (machine learning)*

Instructor sessions: *Categorizing responses based on informed guesses*

Week 9: *Dimensionality Reduction, Principled Component Analysis (PCA)*

Instructor sessions: *What to do with very rich data?*

Week 10: *Outlook, unsupervised classification and neural networks*

Instructor sessions: *Data with no labels and networks*

Week 11: *Organising and preprocessing messy data*

Instructor sessions: *Code review*

Week 12: Final evaluation and wrap-up of course

Instructor sessions: *Ask anything!*

Recap

- Our ways of evaluating models using an explanatory framework are limited when
  - we have no good way of estimating effective parameters
  - we have high degrees of collinearity
  - we have small sample sizes
    - the latter two can lead to unstable models
- Regularisation can improve the stability of a model
  - which comes at the cost of adding bias to the model

Learning goals and outline

Explanation and prediction

1) Understanding how error can be decomposed into bias and variance

2) Understanding when penalised regression may be helpful

3) Understanding how models can be evaluated by out-of-sample testing

Hari R, Puce A (2017) MEG-EEG Primer. Oxford University Press, New York, NY, US

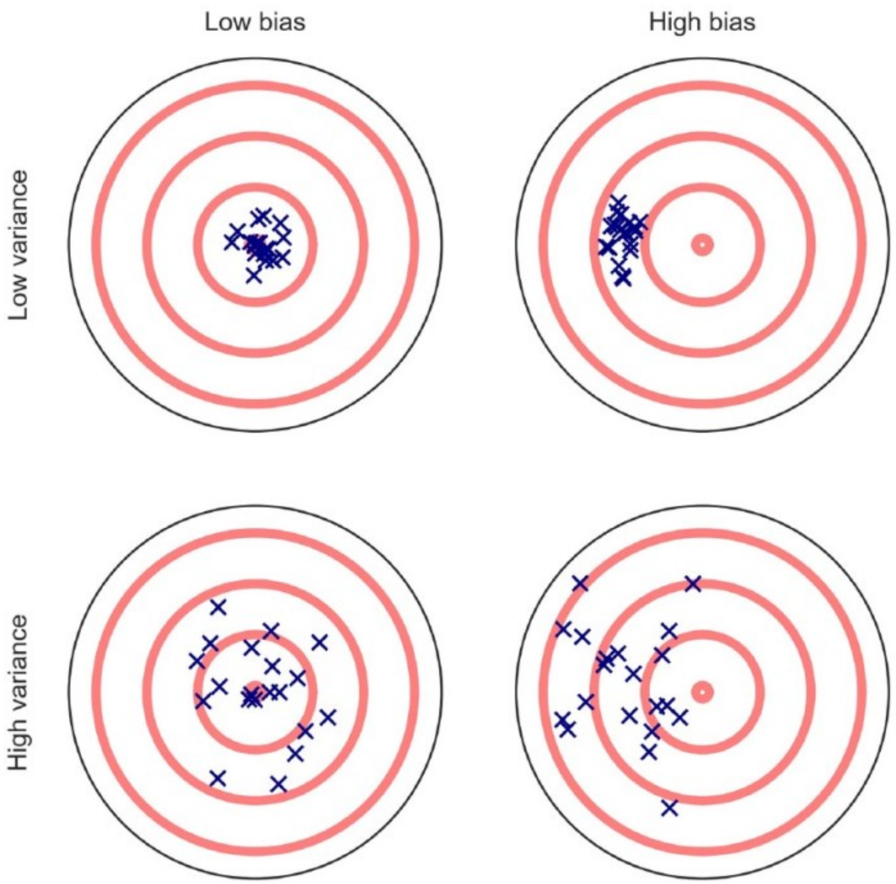


Fig. 2. An estimator's predictions can deviate from the desired outcome (or true scores) in two ways. First, the predictions may display a systematic tendency (or *bias*) to deviate from the central tendency of the true scores (compare right panels with left panels). Second, the predictions may show a high degree of *variance*, or imprecision (compare bottom panels with top panels).

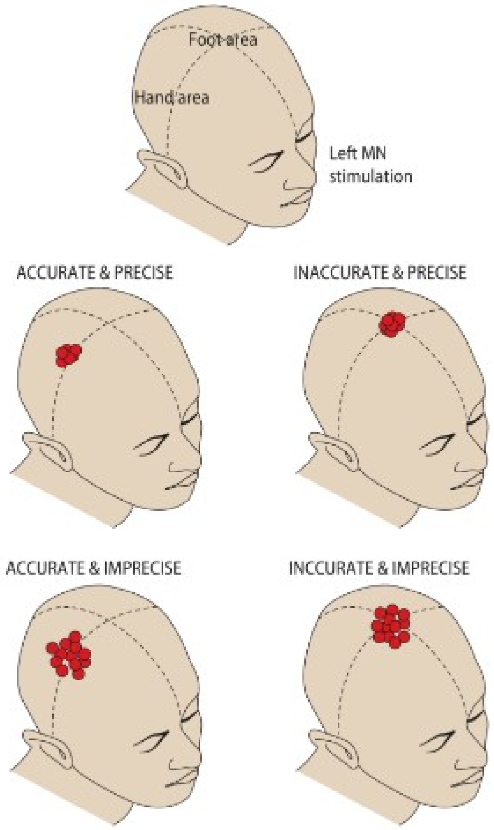


FIGURE 3.7. **Accuracy versus precision.** A schematic illustration of the differences between accuracy and precision of source localization. After left median-nerve stimulation, activations is expected in the right-hemisphere hand region of the primary somatosensory cortex. The foot area is shown at the top of the head. See text for further explanation.

Yarkoni T, Westfall J (2017) Choosing Prediction Over Explanation in Psychology: Lessons From Machine Learning. Perspect Psychol Sci 12:1100–1122.
https://doi.org/10.1177/1745691617693393

Yarkoni T, Westfall J (2017) Choosing Prediction Over Explanation in Psychology: Lessons From Machine Learning. Perspect Psychol Sci 12:1100–1122.
https://doi.org/10.1177/1745691617693393

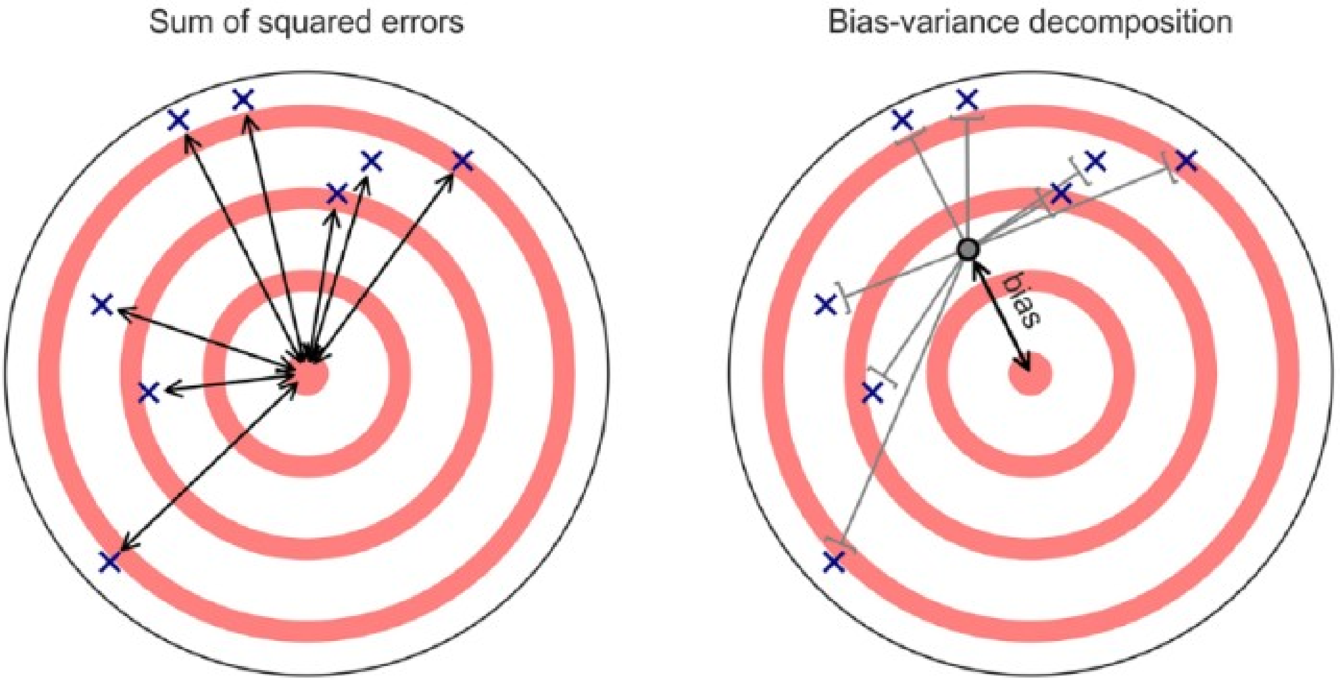


Fig. 3. Schematic illustration of the bias-variance decomposition. (Left) Under the classical error model, prediction error is defined as the sum of squared differences between true scores and observed scores (black lines). (Right) The bias-variance decomposition partitions the total sum of squared errors into two separate components: a bias term that captures a model's systematic tendency to deviate from the true scores in a predictable way (black line) and a variance term that represents the deviations of the individual observations from the model's expected prediction (gray lines).

Bias-variance decomposition

$$MSE = E[(f(x) - \hat{f}(x)^2)] = bias(\hat{f}(x))^2 + var(\hat{f}(x)) + \sigma^2$$

$$bias(\hat{f}(x)) = E[\hat{f}(x)] - f(x)$$

MSE : mean squared error

$E[]$: expected value (often the mean)

$f(x)$: true underlying function

$\hat{f}(x)$: estimated underlying function

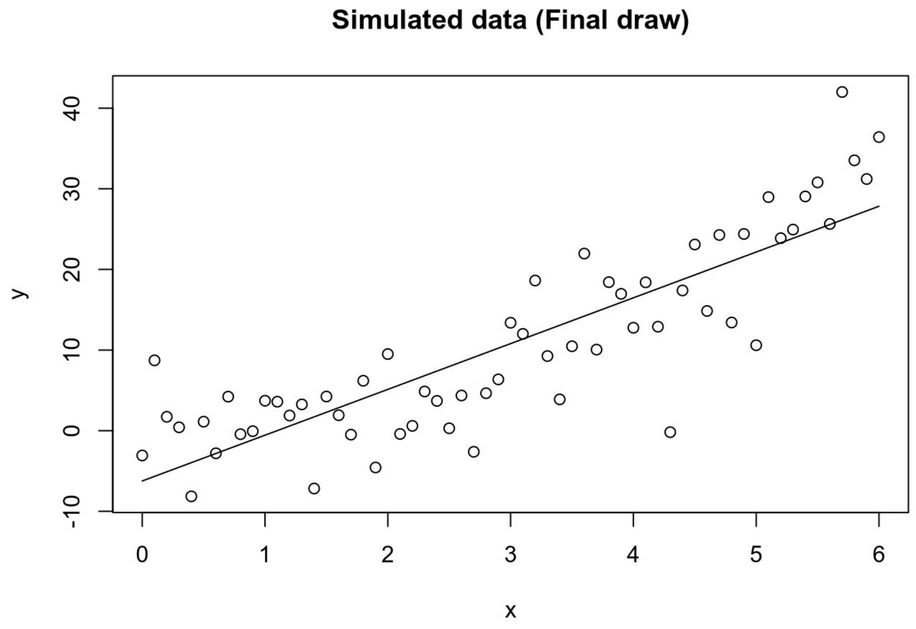
$$MSE = E\left[\left(f(x) - \hat{f}(x)^2\right)\right] = bias\left(\hat{f}(x)\right)^2 + var\left(\hat{f}(x)\right) + \sigma^2$$

$$bias\left(\hat{f}(x)\right) = E\left[\hat{f}(x)\right] - f(x)$$

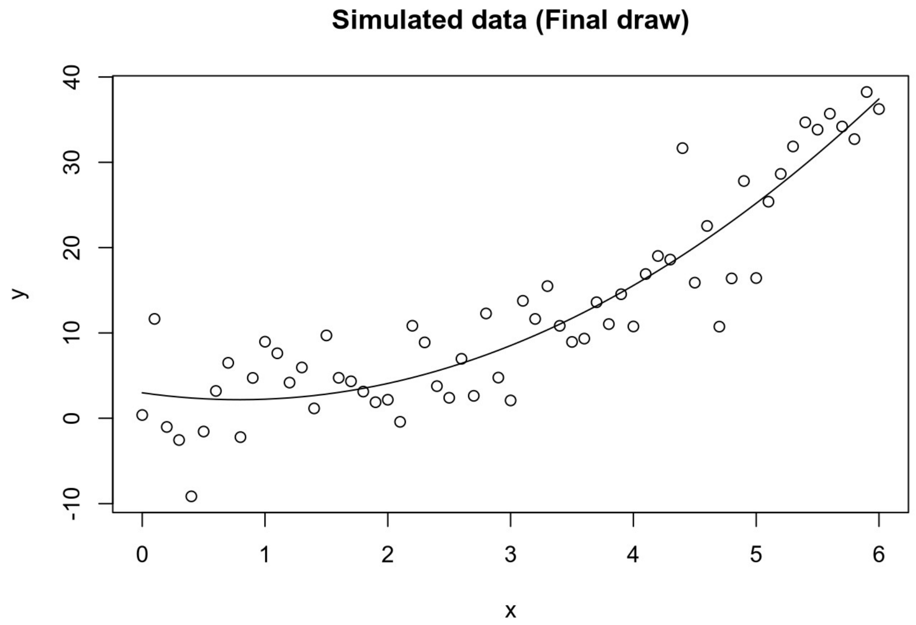
```r
simulate.bias.and.var <- function(formula, fx, x0, x.range, n.sims, sigma)
{
    n.obs <- length(x.range)
    predictions <- matrix(data=NA, nrow=n.obs, ncol=n.sims)
    for(n.sim in 1:n.sims)
    {
        fx.noise <- fx(x.range) + rnorm(n=n.obs, sd=sigma)
        data <- data.frame(x=x.range, y=fx.noise)
        model <- lm(formula, data=data)
        fx.hat <- fitted(model)
        predictions[, n.sim] <- fx.hat
    }

    x.index <- which(x.range == x0)
    eps <- rnorm(n.sims, sd=sigma)
    MSE.estimated <- mean((predictions[x.index, ] - fx(x.range)[x.index] +
                           eps)^2)
    bias <- mean(predictions[x.index, ]) - fx(x.range)[x.index]
    variance <- var(predictions[x.index, ])
    irreducible.error <- var(eps)
    MSE.theoretical <- bias^2 + variance + irreducible.error

    plot(y ~ x, data=data, main='Simulated data (Final draw)')
    lines(x.range, predict(model, newdata=data.frame(x=x.range)))
    print('Difference between estimated and theoretical MSE')
    print(abs(MSE.estimated - MSE.theoretical))
    return(c(bias^2, variance, irreducible.error))
}
```

```r
formulas <- c(
                y ~ x,
                y ~ I(x^2) + x,
                y ~ I(x^3) + I(x^2) + x,
                y ~ I(x^4) + I(x^3) + I(x^2) + x,
                y ~ I(x^5) + I(x^4) + I(x^3) + I(x^2) + x
                )

for(formula in formulas)
{
    error <- simulate.bias.and.var(formula=formula, fx=function(x) x^2, x0=3,
                                    x.range=seq(0, 6, 0.1),
                                     n.sims=1000, sigma=5)

    errors <- rbind(errors, error)
}
```

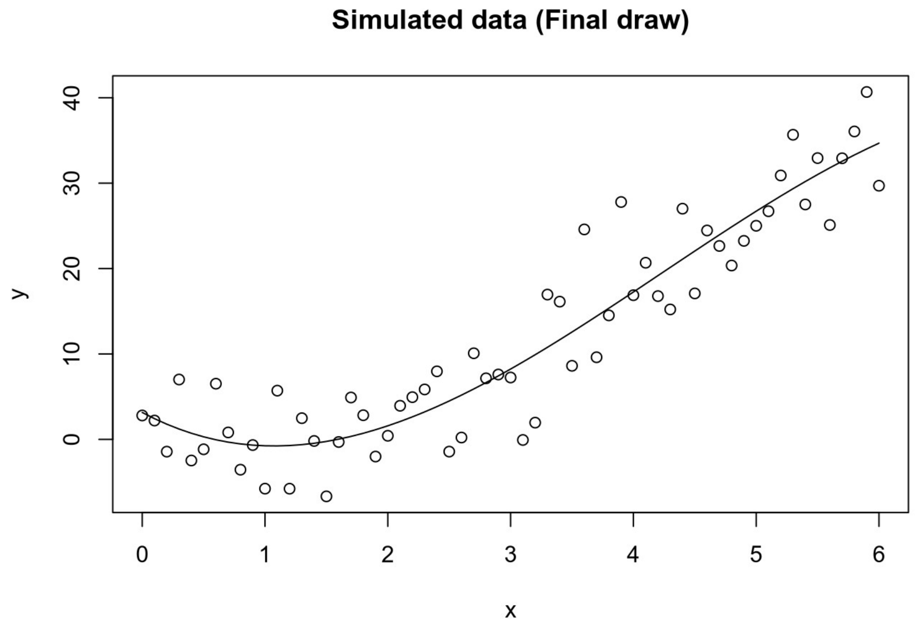
y ~ x



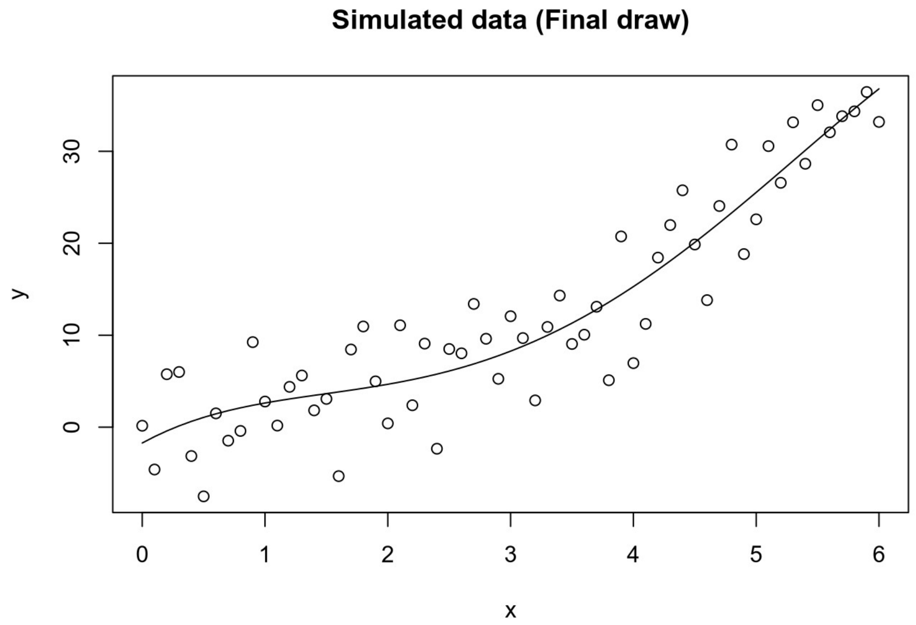
$y \sim x^2 + x$



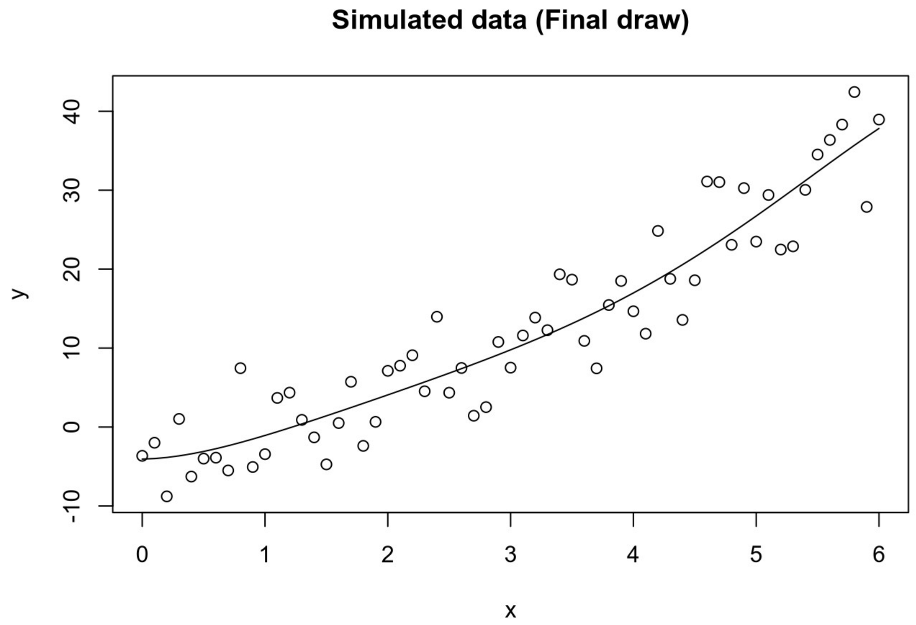
$y \sim x^3 + x^2 + x$



$y \sim x^4 + x^3 + x^2 + x$

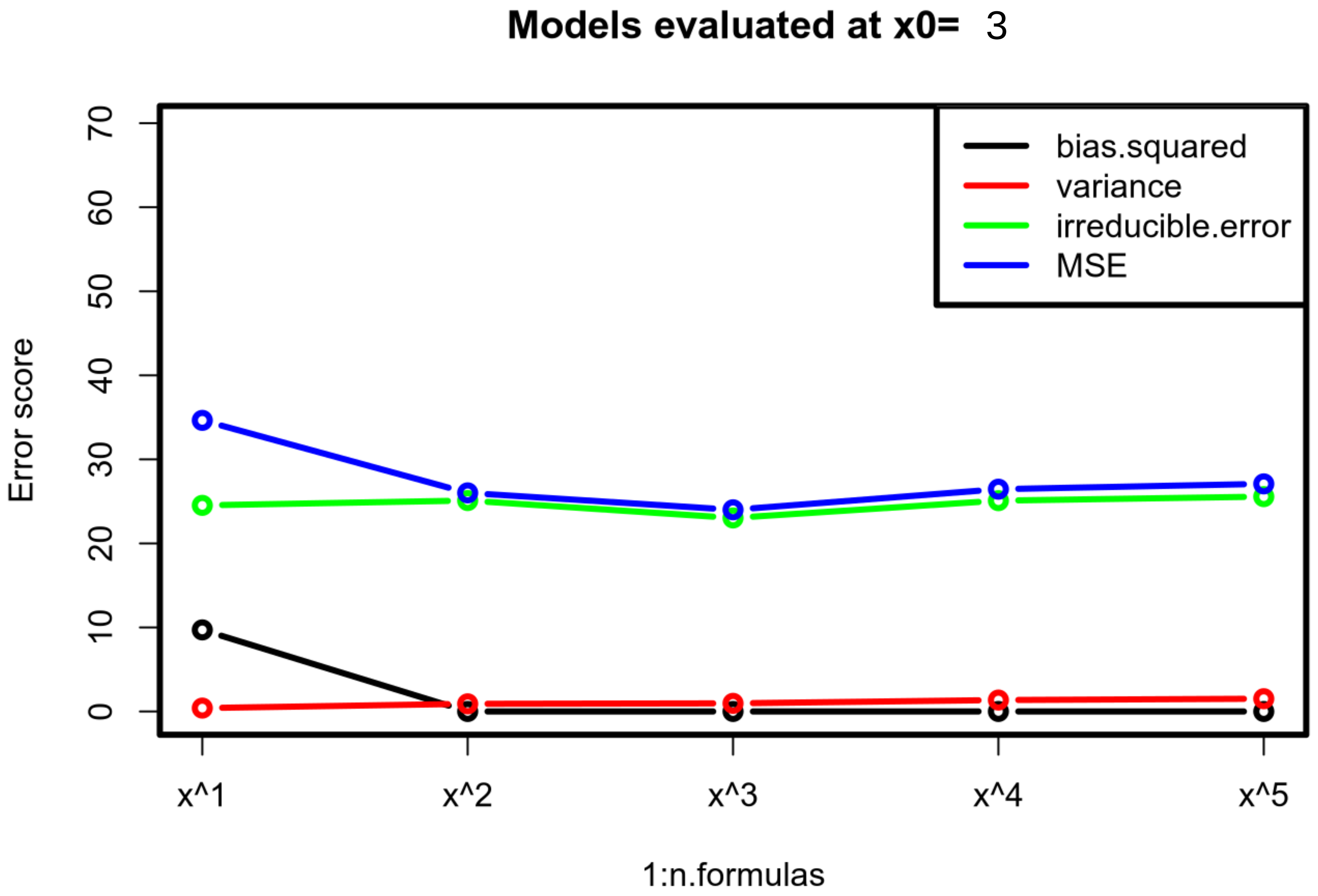


$y \sim x^5 + x^4 + x^3 + x^2 + x$

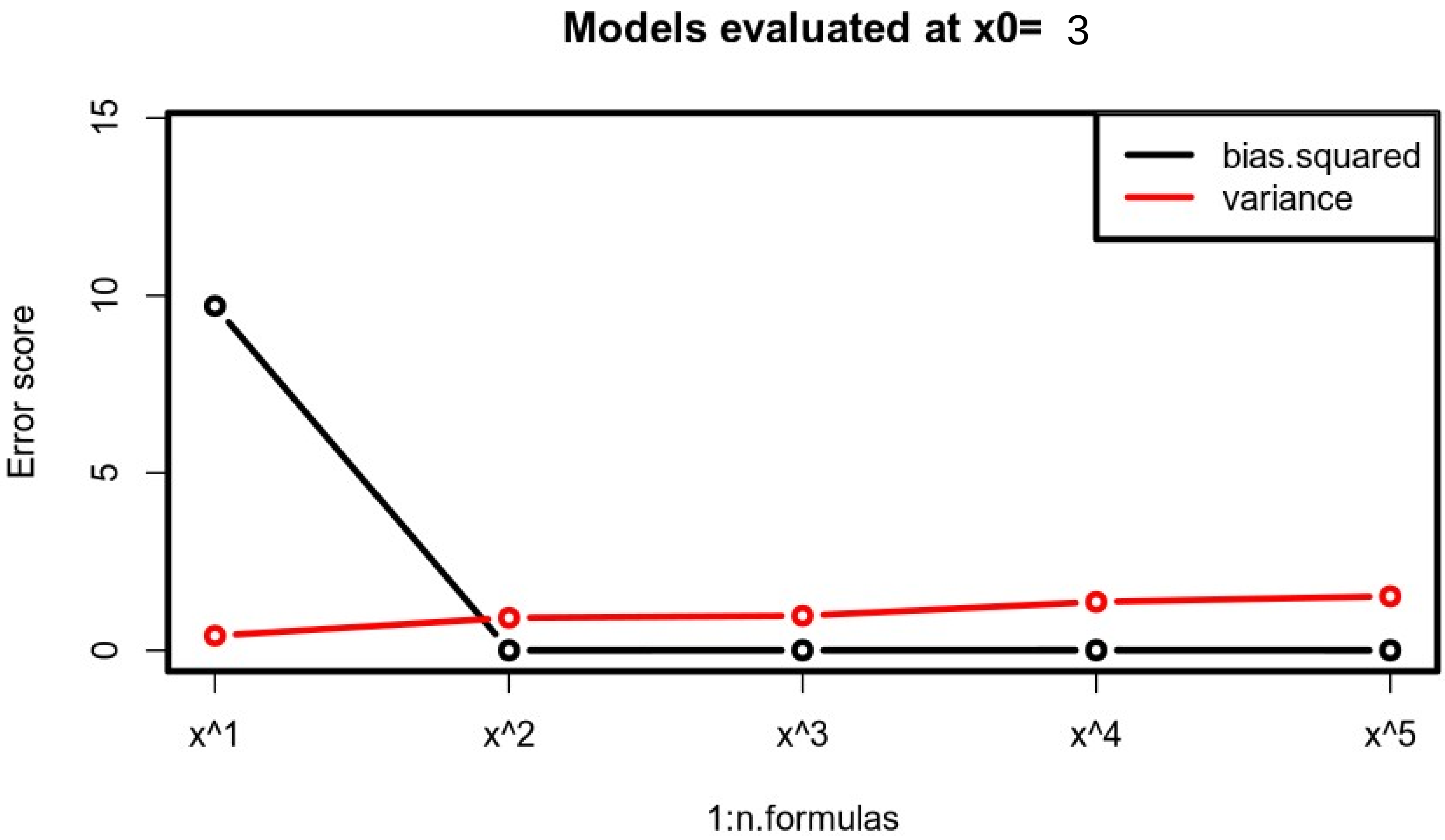


$x_0 = 3$

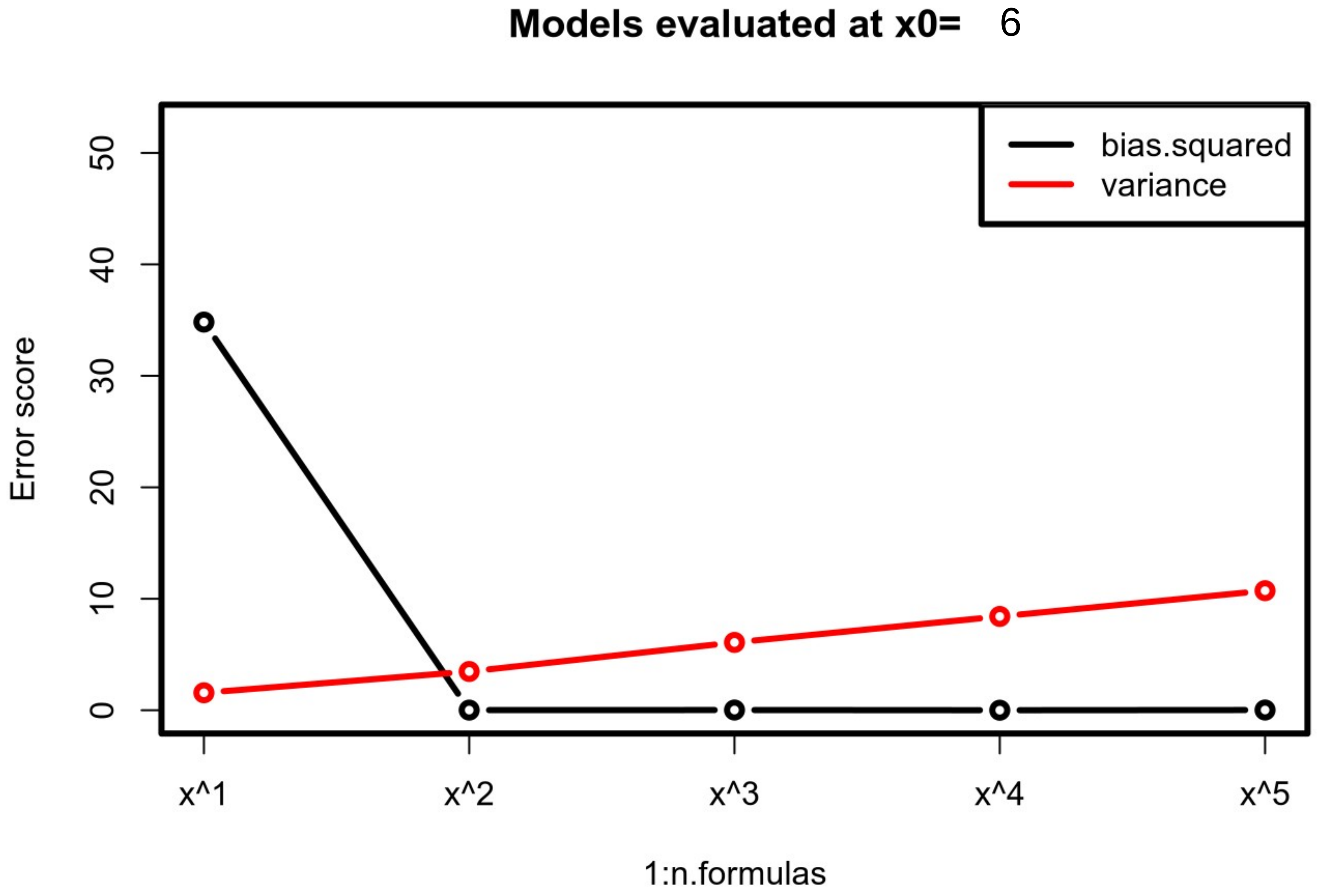
		##	bias.squared	variance	irreducible.error
$y \sim x$	##	1	9.708566e+00	0.4044735	24.52602
$y \sim x^2$ …	##	2	3.096235e-05	0.9119171	25.09829
$y \sim x^3$ …	##	3	1.852917e-03	0.9680085	23.02820
$y \sim x^4$ …	##	4	3.756766e-03	1.3596791	25.08635
$y \sim x^5$ …	##	5	1.799701e-04	1.5190037	25.56529

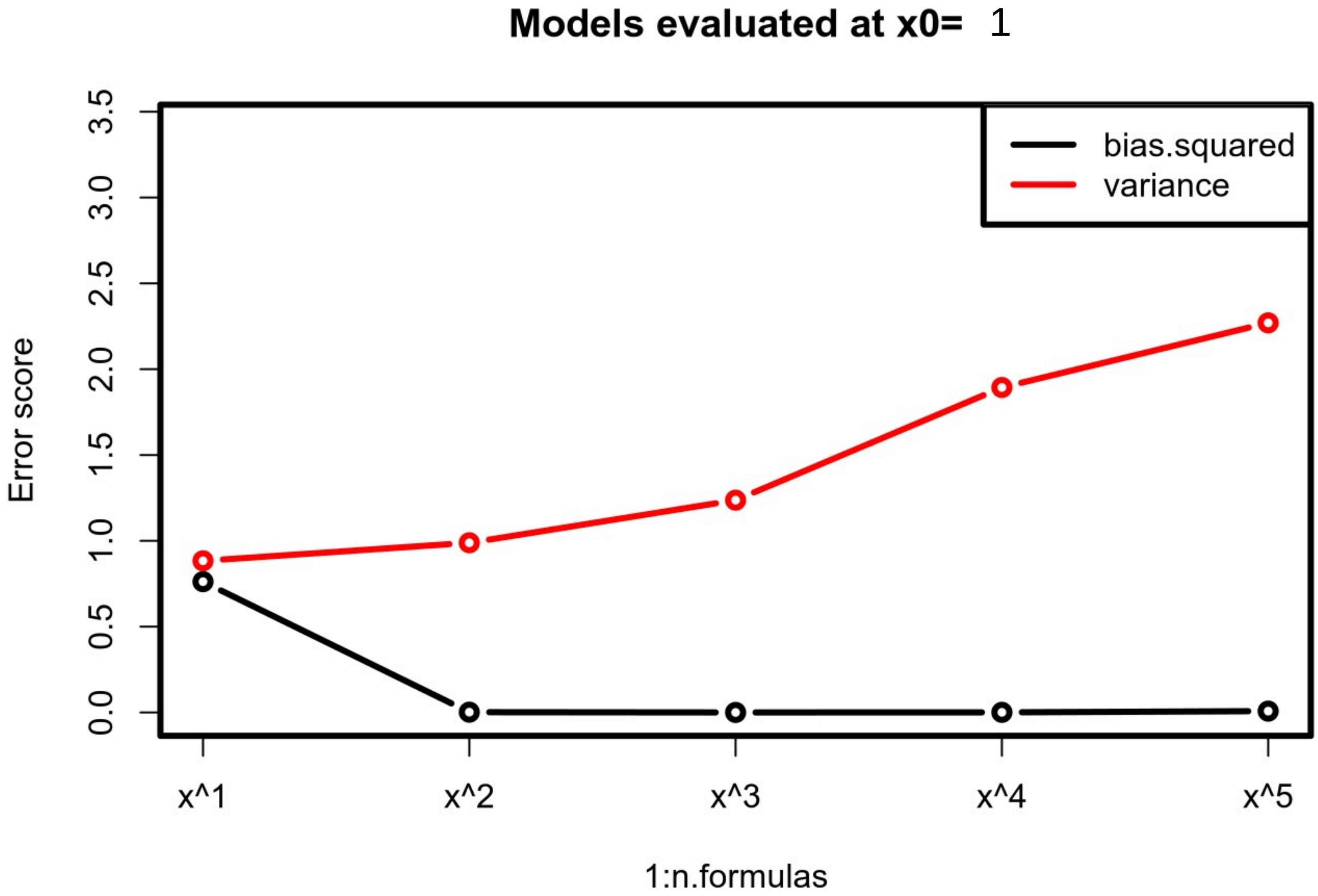


Models evaluated at x0= 3
Error score
0
10
20
30
40
50
60
70
bias.squared
variance
irreducible.error
MSE
x^1
x^2
x^3
x^4
x^5
1:n.formulas


Models evaluated at x0= 3
bias.squared
variance
Error score
0
5
10
15
x^1
x^2
x^3
x^4
x^5
1:n.formulas

Discussion points:
What happens to variance when x_0 is increased?
(say to 6)
What happens to variance when x_0 is decreased?
(say to 1)


Models evaluated at x0= 6
bias.squared
variance
Error score
0
10
20
30
40
50
x^1
x^2
x^3
x^4
x^5
1:n.formulas

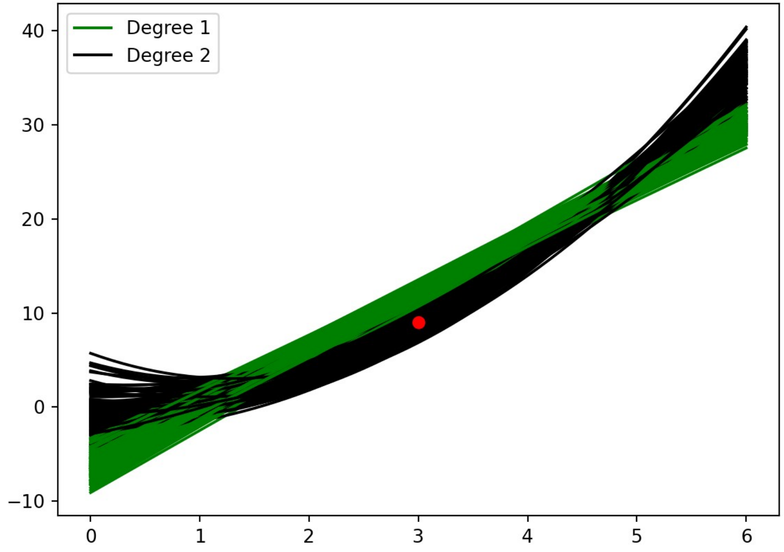

Models evaluated at x0= 1
bias.squared
variance
Error score
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
x^1
x^2
x^3
x^4
x^5
1:n.formulas

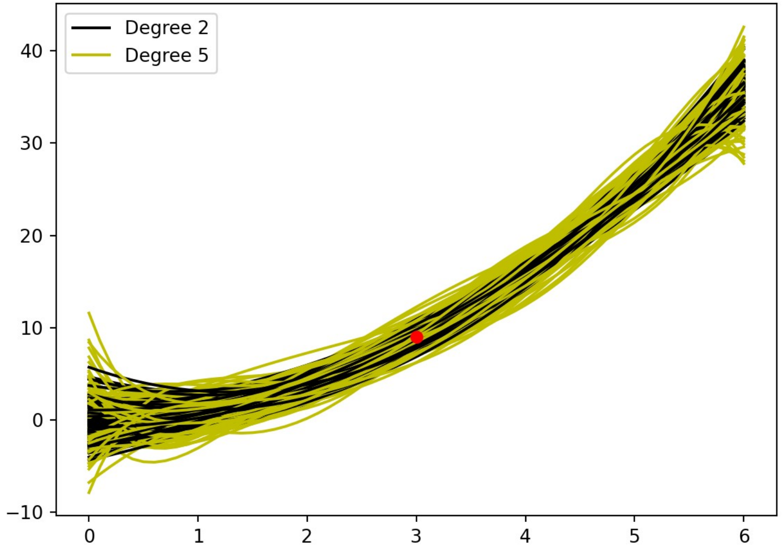
$x_0 = 1$	bias.squared	variance
1	0.7620209937	0.8835371
2	0.0017383448	0.9885182
3	0.0001546774	1.2368800
4	0.0007538773	1.8933452
5	0.0078616986	2.2698603

$x_0 = 3$	bias.squared	variance
1	9.708566e+00	0.4044735
2	3.096235e-05	0.9119171
3	1.852917e-03	0.9680085
4	3.756766e-03	1.3596791
5	1.799701e-04	1.5190037

$x_0 = 6$	bias.squared	variance
1	34.821206760	1.565046
2	0.012470904	3.474977
3	0.018714117	6.082202
4	0.002293859	8.414034
5	0.019602168	10.724150

Simulations for polynomials of degree 1, 2 and 5 at $x_0 = 3$





OUT OF SAMPLE TESTING
as a way of evaluating models

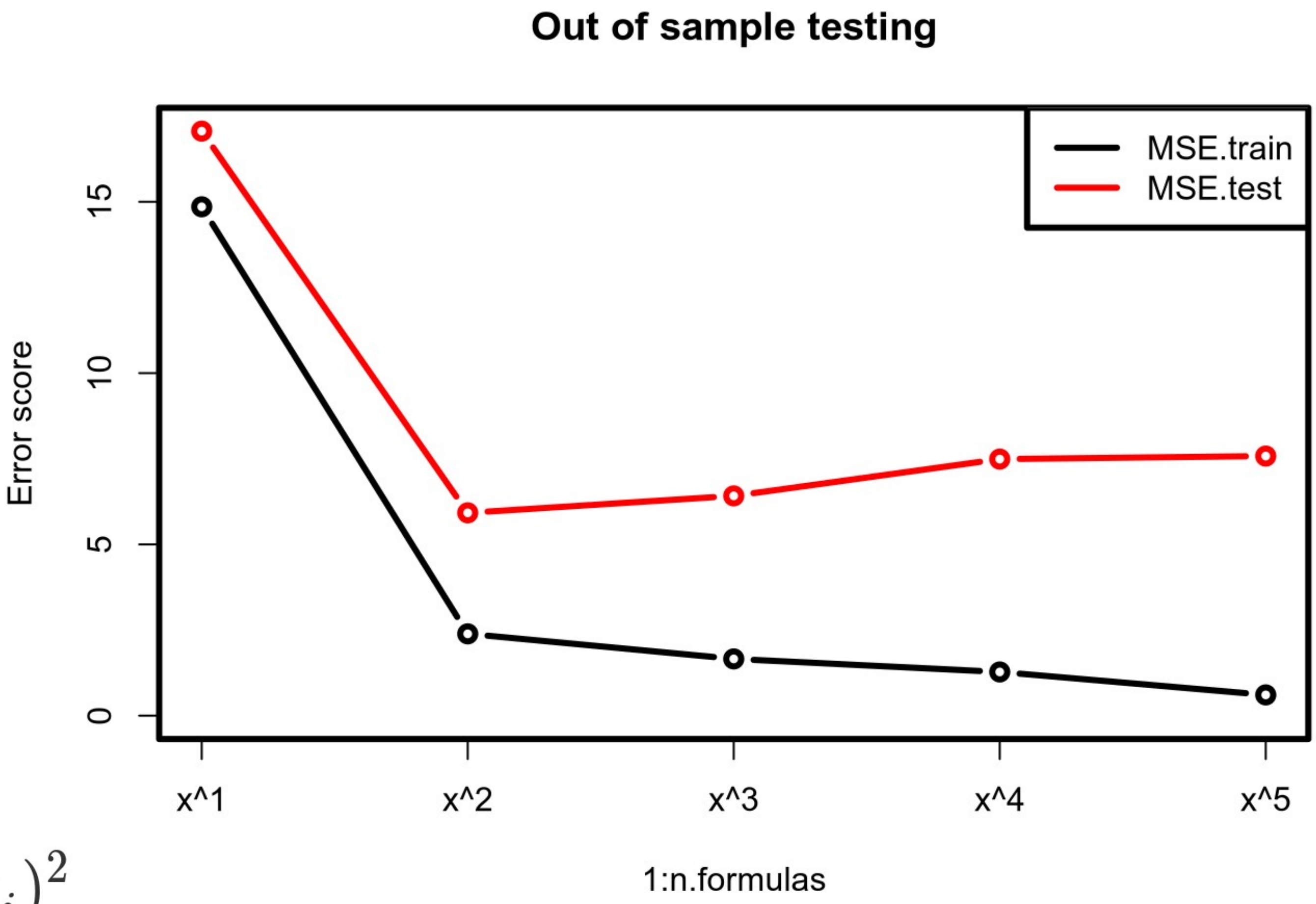
Using mean squared error as a measure of predictive power

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

```r
predict.out.of.sample <- function(x, fx, formula, sigma, n.sims)
{
    n.obs <- length(x)
    MSE.trains <- numeric(n.sims)
    MSE.tests  <- numeric(n.sims)
    for(n.sim in 1:n.sims)
    {
        fx.train <- fx(x) + rnorm(n.obs, sd=sigma)
        fx.test  <- fx(x) + rnorm(n.obs, sd=sigma)

        data.train <- data.frame(x=x, y=fx.train)

        model <- lm(formula, data=data.train)
        fx.hat.train <- fitted(model)
        MSE.trains[n.sim] <- 1/n.obs * sum((fx.train - fx.hat.train)^2)
        MSE.tests[n.sim]   <- 1/n.obs * sum((fx.test - fx.hat.train)^2)
    }
    print(formula)
    print(paste('MSE train:', round(mean(MSE.trains), 2)))
    print(paste('MSE test:',  round(mean(MSE.tests), 2)))
    return(c(mean(MSE.trains), mean(MSE.tests)))
}
```

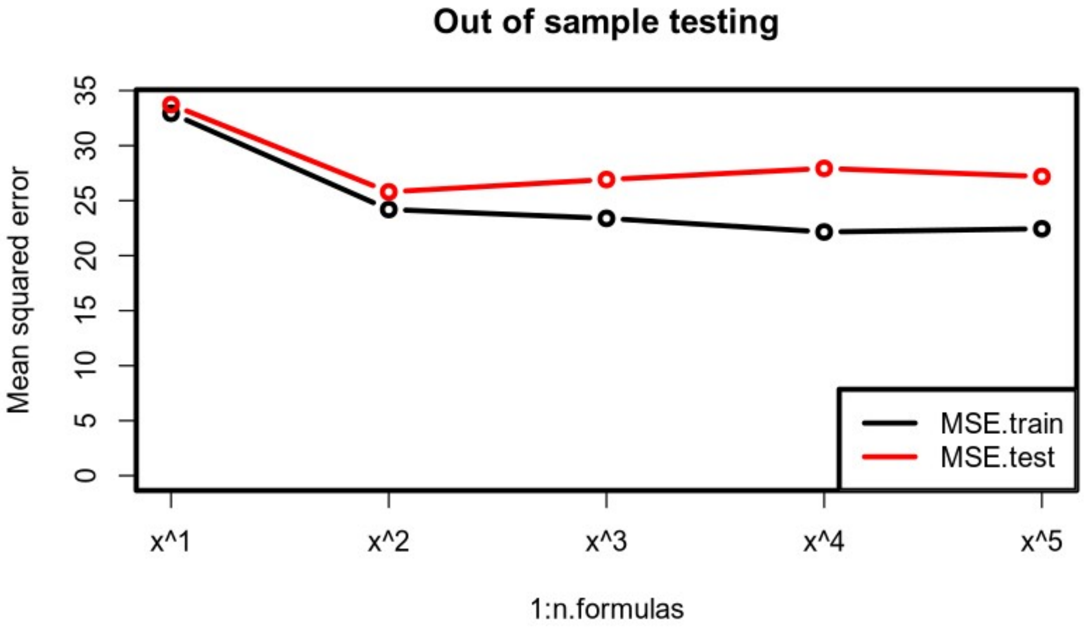
```r
formulas <- c(
                y ~ x,
                y ~ I(x^2) + x,
                y ~ I(x^3) + I(x^2) + x,
                y ~ I(x^4) + I(x^3) + I(x^2) + x,
                y ~ I(x^5) + I(x^4) + I(x^3) + I(x^2) + x
                )
n.formulas <- length(formulas)
MSEs <- matrix(data=NA, nrow=2, ncol=n.formulas)
for(formula.index in 1:n.formulas)
{
    formula <- formulas[[formula.index]]
    MSEs[, formula.index] <- predict.out.of.sample(x=seq(0, 6, 1),
                                                        fx=function(x) x^2,
                                          formula=formula, sigma=2, n.sims=100)
}
```

$\sigma = 2$
$n.obs = 7$



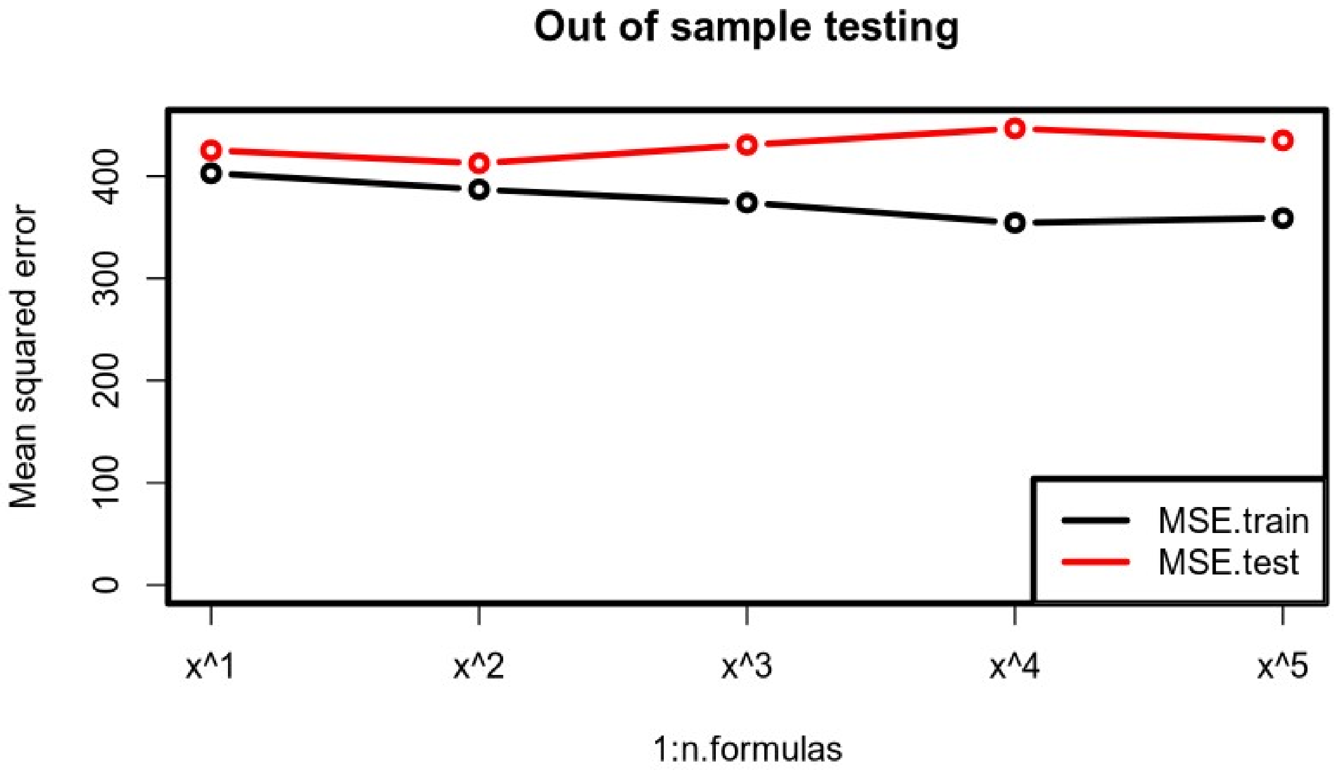
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$\sigma = 5$
$n.obs = 61$



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

$\sigma = 20$
$n.obs = 61$



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

Interim summary

So if greater variance is detrimental to prediction, we may be able improve prediction by introducing bias (and thereby reducing variance)

Penalised regression

$$\text{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \left(\text{minimise to obtain least squares solution}\right)$$

$$\text{lasso regression}: \text{RSS} + \lambda \sum_{j=1}^{p} |\beta_j| \left(\text{minimise this sum}\right)$$

$$\text{ridge regression}: \text{RSS} + \lambda \sum_{j=1}^{p} (\beta_j^2) \left(\text{minimise this sum}\right)$$

n: number of observations
p: number of predictor variables
λ: a constant

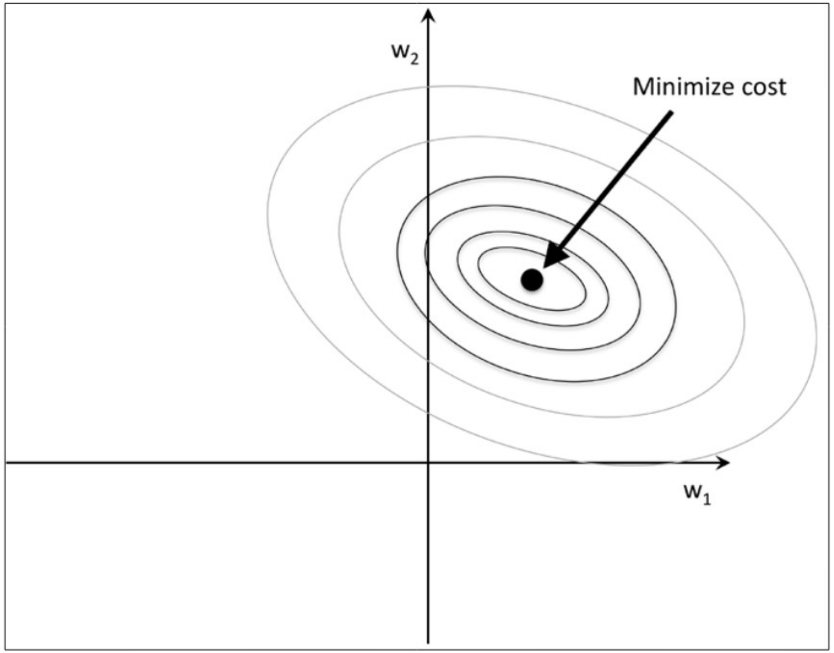
Group discussion

In each case: what happens when?

1. λ increases?
2. λ decreases?
3. λ is 0?
4. λ goes towards infinity?

Least squares

$$J(w) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$



p. 113; Raschka S (2015) Python Machine Learning. Packt Publishing Ltd

L2 regularization, ridge

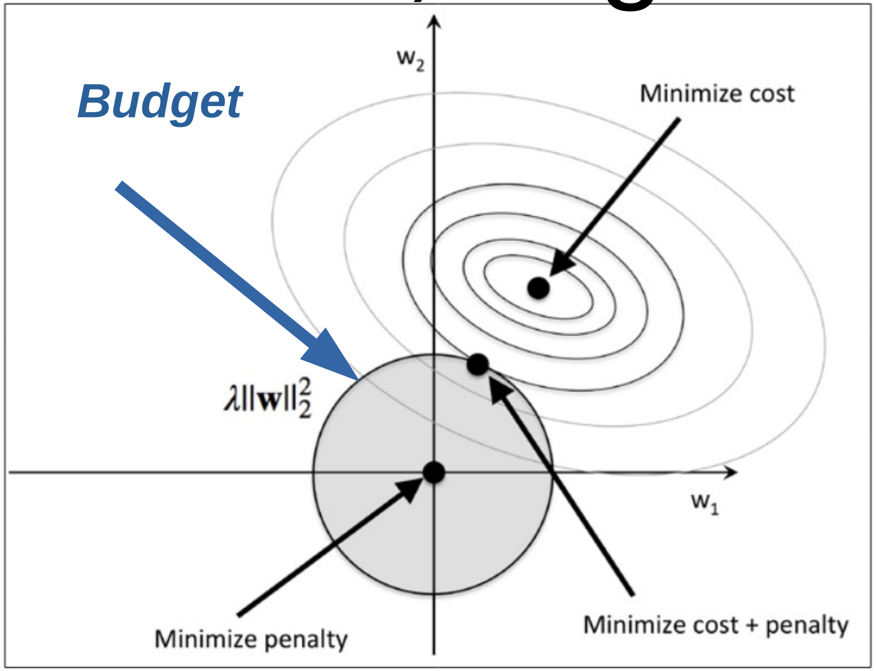
Why is the *budget* round?

Compare with a circle centred at (0,0)

$$x^2 + y^2 = r^2$$

$$w_1^2 + w_2^2 = r^2$$

$$\|w\|_2 = \sqrt{(w_1^2 + w_2^2)}$$



(p. 114: Raschka, 2015)

$$J(w)_{Ridge} = \sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 + \lambda \| w \|_2^2$$

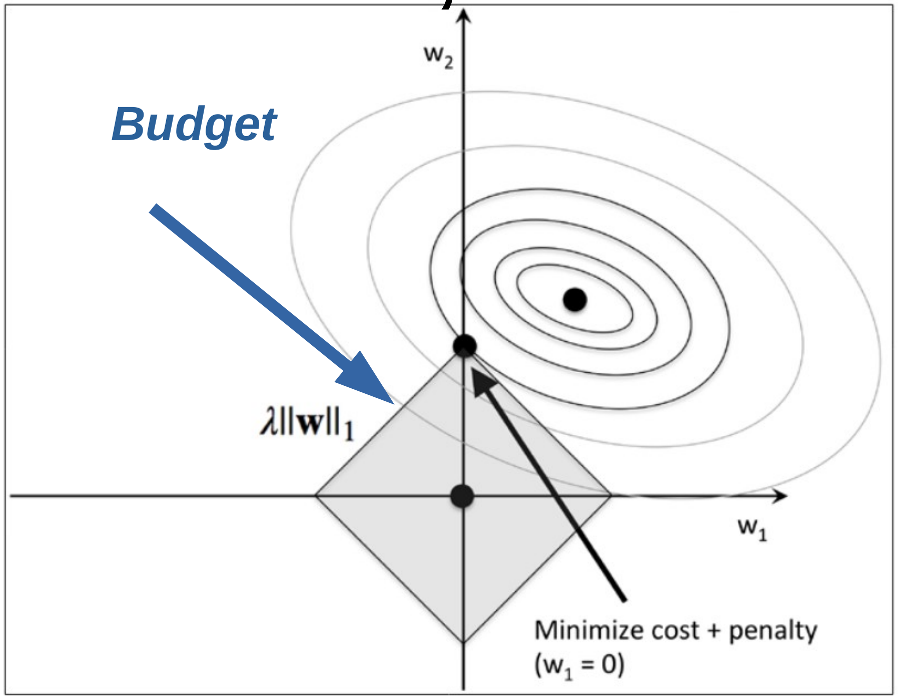
p. 115; Raschka S (2015) Python Machine Learning. Packt Publishing Ltd

L1 regularization, lasso

Why is the *budget* square?

$$\|w\|_1 = |w_1| + |w_2|$$
$$\text{if } w_1 = max(w_1) \text{ then } w_2 = 0$$
$$\text{if } w_2 = max(w_2) \text{ then } w_1 = 0$$

(p. 115: Raschka, 2015)



$$J(w)_{LASSO} = \sum_{i=1}^{n} \left(y^{(i)} - \hat{y}^{(i)}\right)^2 + \lambda \| w \|_1$$

Out-of-sample as validity check


```r
mtcars.1 <- mtcars[1:10, ]
```

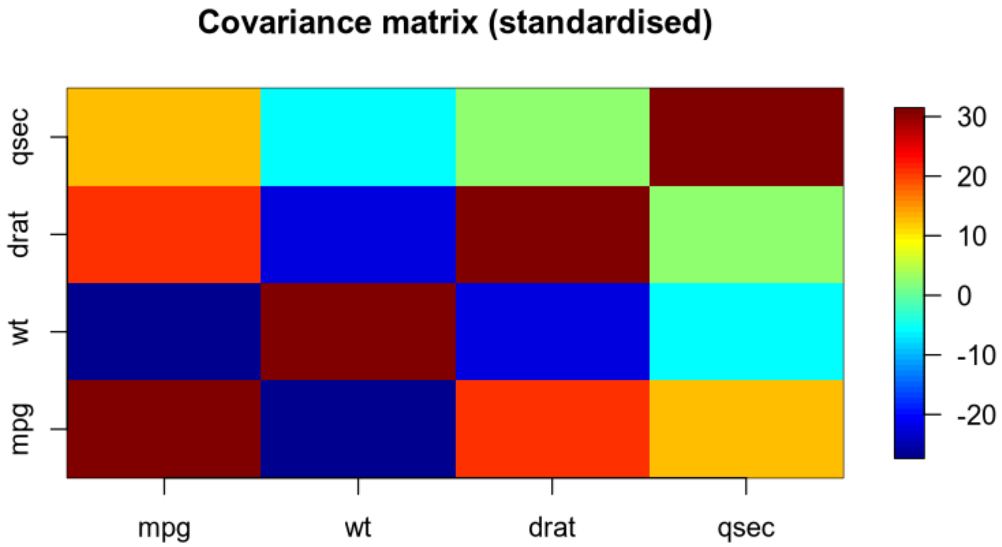
Out-of-sample as validity check

```
Call:
lm(formula = hp ~ mpg + wt + drat + qsec, data = mtcars.1)

Coefficients:
(Intercept)          mpg           wt         drat         qsec
    414.541      -13.638       12.753       11.263       -5.042
```

```
Call:
lm(formula = hp ~ mpg + wt + drat + qsec, data = mtcars)

Coefficients:
(Intercept)          mpg           wt         drat         qsec
    473.779       -2.877       26.037        4.819      -20.751
```

Collinearity

MTCARS

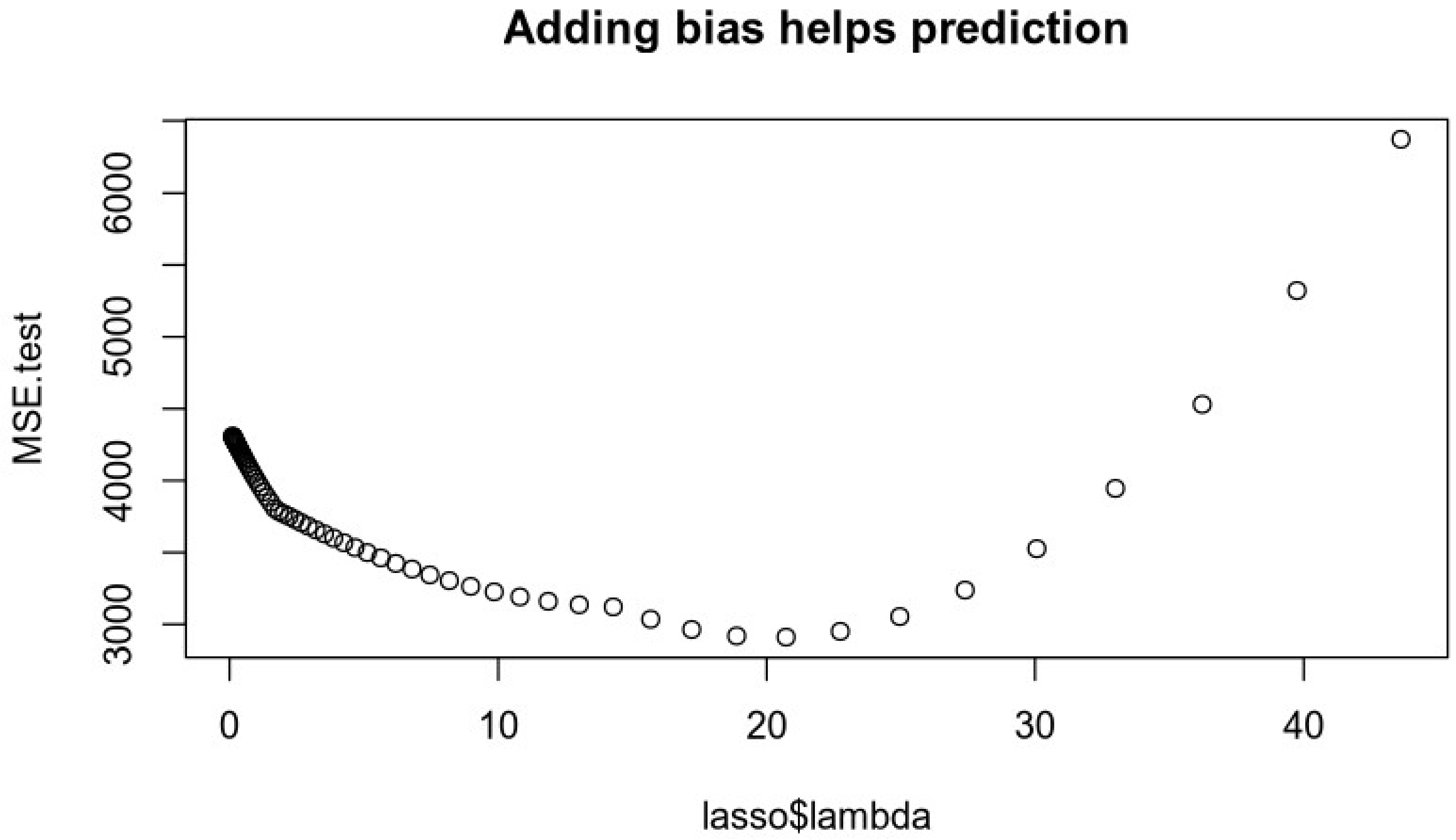


Suddenly, someone shows up with

Let's check our model

```r
mtcars.2 <- mtcars[11:32, ]
```

```r
rm(list =ls())
library(glmnet)
y.train <- as.matrix(mtcars[1:10, 4]) ## hp
x.train <- as.matrix(mtcars[1:10, c(1, 6, 5, 7)]) # mpg, wt, drat, qsec
y.test <- as.matrix(mtcars[11:32, 4]) ## hp
x.test <- as.matrix(mtcars[11:32, c(1, 6, 5, 7)])# mpg, wt, drat, qsec

lasso <- glmnet(x=x.train, y=y.train, alpha=1) ## get RSS and penalty in
n.models <- lasso$dim[2]
MSE.train <- numeric(n.models)
penalty.train <- numeric(n.models)
MSE.test <- numeric(n.models)
for(model.index in 1:n.models)
{
    betas <- c(lasso$a0[model.index], lasso$beta[, model.index])
    X.train <- cbind(1, x.train)
    X.test  <- cbind(1, x.test)
    y.hat.train <- X.train %*% betas
    y.hat.test  <- X.test %*% betas
    MSE.train[model.index] <- 1 / length(y.train) * sum((y.train - y.hat.train)^2)
    penalty.train[model.index] <- sum(abs(lasso$beta[, model.index]))
    MSE.test[model.index] <- 1/length(y.test) * sum((y.test - y.hat.test)^2)
}

plot(lasso$lambda, MSE.test)
```

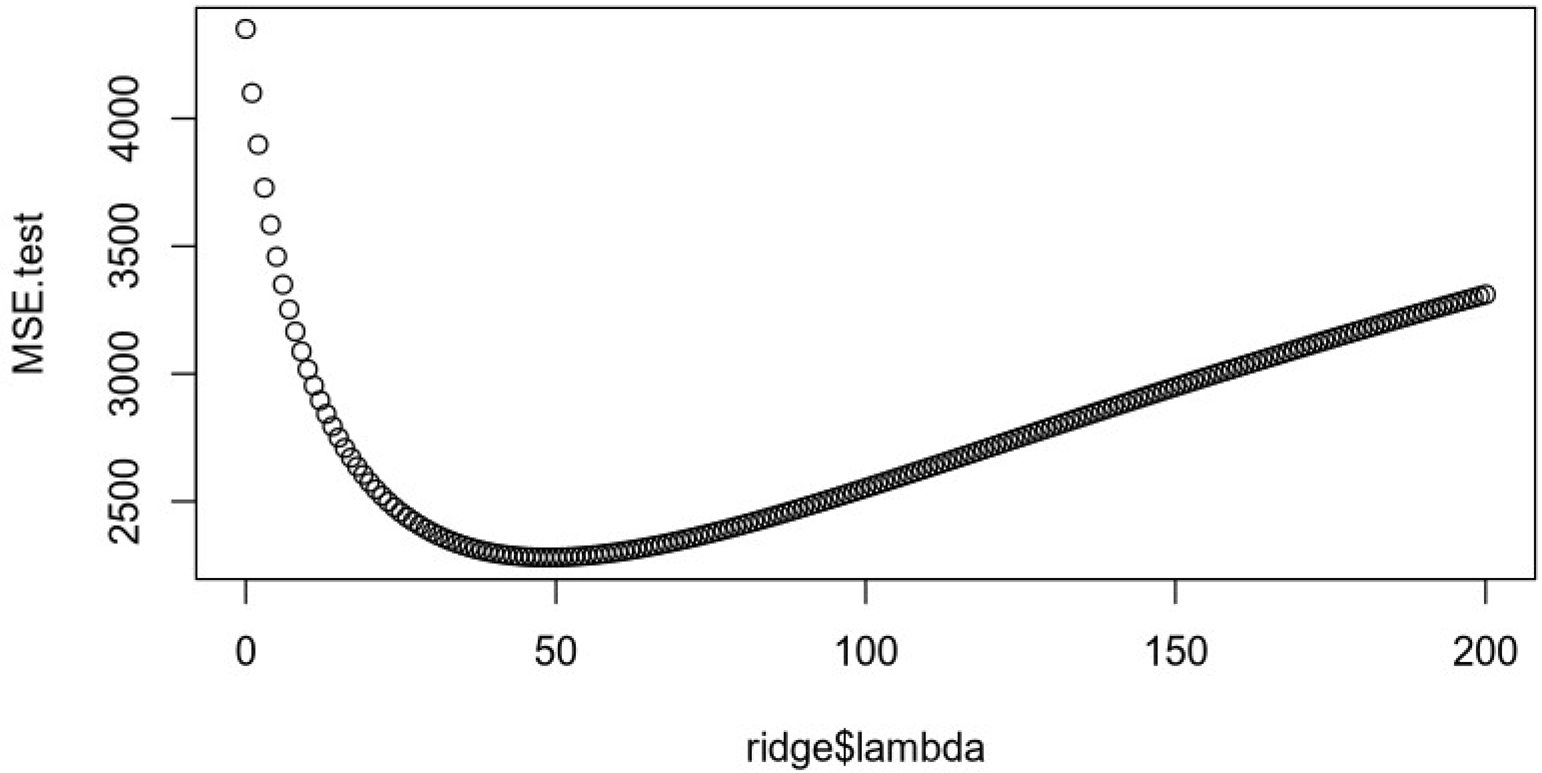


Adding bias helps prediction
MSE.test
3000
4000
5000
6000
0
10
20
30
40
lasso$lambda

lambda <dbl>	MSE.train <dbl>	penalty.train <dbl>	MSE.test <dbl>
43.6220702	2382.3600	0.000000	6371.040
39.7468057	2059.2836	1.405142	5321.815
36.2158090	1791.0598	2.685456	4530.285
32.9984963	1568.3758	3.852030	3945.631
30.0670009	1383.4995	4.914968	3526.290
27.3959314	1230.0121	5.883478	3238.326
24.9621524	1102.5840	6.765949	3054.088
22.7445835	996.7911	7.570023	2951.093
20.7240174	908.9599	8.302665	2911.110
18.8829528	836.0409	8.970222	2919.395

```
glmnet(x = x.train, y = y.train, alpha = 1)

       s8        mpg         wt       drat       qsec
291.925294  -8.302665   0.000000   0.000000   0.000000

Call:
lm(formula = hp ~ mpg + wt + drat + qsec + 1, data = mtcars[1:10, ])
Coefficients:
(Intercept)          mpg           wt         drat         qsec
    414.541      -13.638       12.753       11.263       -5.042
```

```r
## ridge
rm(list =ls())
library(glmnet)
y.train <- as.matrix(mtcars[1:10, 4]) ## hp
x.train <- as.matrix(mtcars[1:10, c(1, 6, 5, 7)]) # mpg, wt, drat, qsec
y.test <- as.matrix(mtcars[11:32, 4]) ## hp
x.test <- as.matrix(mtcars[11:32, c(1, 6, 5, 7)])# mpg, wt, drat, qsec

ridge <- glmnet(x=x.train, y=y.train, alpha=0, lambda=0:200)
n.models <- ridge$dim[2]
MSE.train <- numeric(n.models)
penalty.train <- numeric(n.models)
MSE.test <- numeric(n.models)
for(model.index in 1:n.models)
{
    betas <- c(ridge$a0[model.index], ridge$beta[, model.index])
    X.train <- cbind(1, x.train)
    X.test  <- cbind(1, x.test)
    y.hat.train <- X.train %*% betas
    y.hat.test  <- X.test %*% betas
    MSE.train[model.index] <- 1 / length(y.train) * sum((y.train - y.hat.train)^2)
    penalty.train[model.index] <- sum(ridge$beta[, model.index]^2)
    MSE.test[model.index] <- 1/length(y.test) * sum((y.test - y.hat.test)^2)
}

plot(ridge$lambda, MSE.test, main='Adding bias helps prediction')
```

Adding bias helps prediction



lambda <dbl>	MSE.train <dbl>	penalty.train <dbl>	MSE.test <dbl>
50	711.7847	409.7758	2280.286
49	705.3788	413.2714	2279.970
48	698.9470	416.7924	2280.049
47	692.4899	420.3382	2280.541
46	686.0082	423.9078	2281.469
45	679.5026	427.5003	2282.853
44	672.9740	431.1146	2284.718
43	666.4231	434.7496	2287.087
42	659.8510	438.4039	2289.987
41	653.2585	442.0760	2293.446

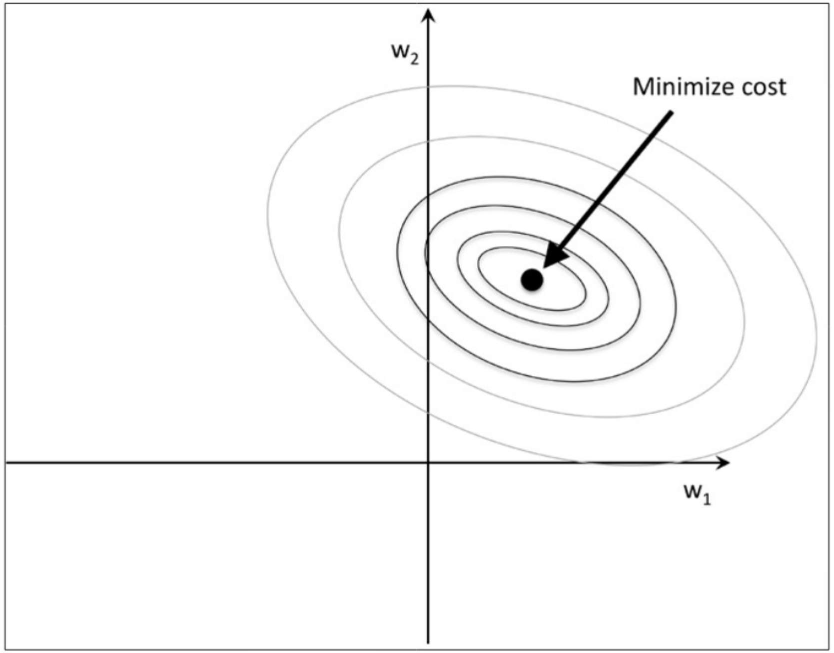
```
glmnet(x = x.train, y = y.train, alpha = 0, lambda = 0:200)
      s151        mpg         wt       drat       qsec
312.148253  -5.788459  17.325500  -7.093469  -5.410624

Call:
lm(formula = hp ~ mpg + wt + drat + qsec + 1, data =
mtcars[1:10, ])
Coefficients:
(Intercept)          mpg           wt         drat         qsec
    414.541      -13.638       12.753       11.263       -5.042
```

Adding bias to a model can increase stability of the model and can in turn increase prediction capability

Remember in linear regression MSE is a metric for prediction error

Least squares

$$J(w) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$



p. 113; Raschka S (2015) Python Machine Learning. Packt Publishing Ltd

L2 regularization, ridge

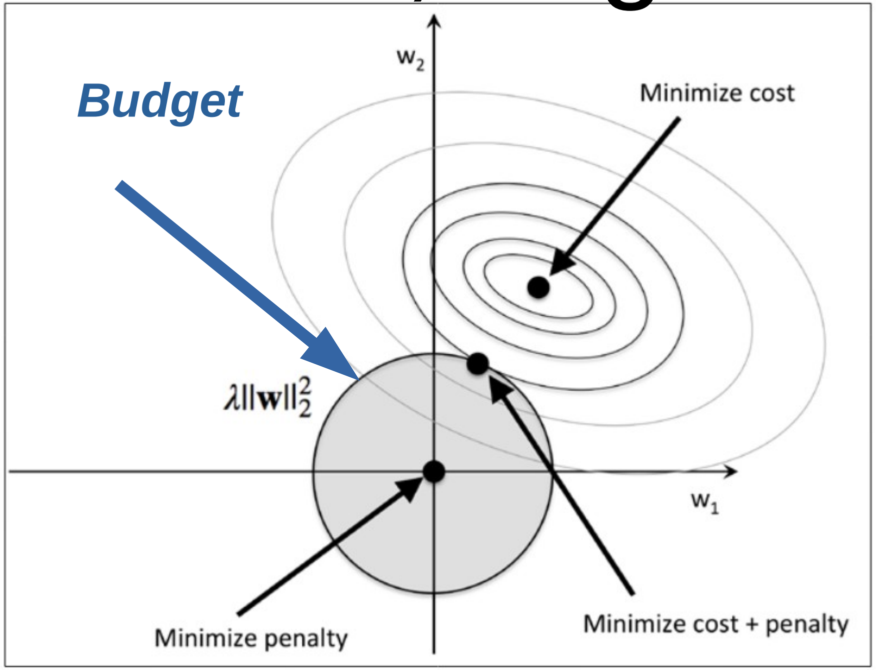
Why is the *budget* round?

Compare with a circle centred at (0,0)

$$x^2+y^2=r^2$$

$$w_1^2+w_2^2=r^2$$

$$\|w\|_2=\sqrt{(w_1^2+w_2^2)}$$



(p. 114: Raschka, 2015)

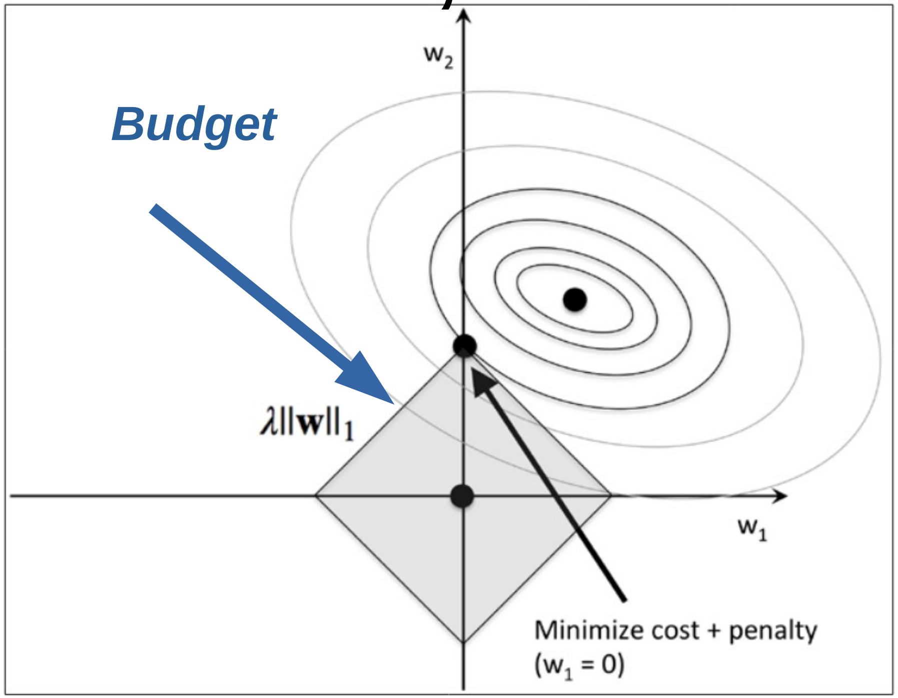
$$J(w)_{Ridge}=\sum_{i=1}^{n}\left(y^{(i)}-\hat{y}^{(i)}\right)^2+\lambda\|w\|_2^2$$

p. 115; Raschka S (2015) Python Machine Learning. Packt Publishing Ltd

L1 regularization, lasso

Why is the *budget* square?

$$\|w\|_1 = |w_1| + |w_2|$$
$$\text{if } w_1 = max(w_1) \text{ then } w_2 = 0$$
$$\text{if } w_2 = max(w_2) \text{ then } w_1 = 0$$



(p. 115: Raschka, 2015)

$$J(w)_{LASSO} = \sum_{i=1}^{n} \left(y^{(i)} - \hat{y}^{(i)}\right)^2 + \lambda \| w \|_1$$

We can combine L1 and L2

JUST FOR COMPLETION

$$J(w)_{ElasticNet} = \sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 + \lambda_1 \sum_{j=1}^{m} w_j{}^2 + \lambda_2 \sum_{j=1}^{m} \left|w_j\right|$$

Summary

- Bias can be added in ways that improve prediction
  - it does this by removing collinearity
  - thereby making the model stable
  - and more generalisable
- This is one backbone of machine learning

Learning goals and outline

Explanation and prediction

1) Understanding how error can be decomposed into bias and variance

2) Understanding when penalised regression may be helpful

3) Understanding how models can be evaluated by out-of-sample testing

The course plan

Week 1: *Introduction*

Instructor sessions: *Setting up* R *and* Python *and recollection of the general linear model*

Week 2: *Multilevel linear regression*

Instructor sessions: *Modelling subject level effects – and how do they differ from group level effects?*

Week 3: *Link functions and fitting generalised linear multilevel models*

Instructor sessions: *What to do when the response variable is not continuous?*

Week 4: *Evaluating Generalised linear mixed models*

Instructor sessions: *How do we assess how models compare to one another?*

Week 5: *Explanation and Prediction*

Instructor sessions: *Code review*

Week 6: *Mid-way evaluation and Machine Learning Intro*

Instructor sessions: *Getting Python Running*

Week 7: *Linear regression revisited (machine learning)*

Instructor sessions: *How to constrain our models to make them more predictive*

Week 8: *Logistic regression revisited (machine learning)*

Instructor sessions: *Categorizing responses based on informed guesses*

Week 9: *Dimensionality Reduction, Principled Component Analysis (PCA)*

Instructor sessions: *What to do with very rich data?*

Week 10: *Outlook, unsupervised classification and neural networks*

Instructor sessions: *Data with no labels and networks*

Week 11: *Organising and preprocessing messy data*

Instructor sessions: *Code review*

Week 12: Final evaluation and wrap-up of course

Instructor sessions: *Ask anything!*

Next time

- Introduction to classification
  - The Perceptron
  - ADAline
- Linear regression in machine learning
  - Looking at big(ger) scale data

Reading questions

- Chapter 1
  - What are the differences between supervised, unsupervised and reinforcement learning?
  - What is the difference between classification and regression?
  - What is dimensionality reduction?
    - Is it similar to regularisation?
- Chapter 2
  - How does the w vector and $\boldsymbol{x}$ matrix relation to what we know as $\boldsymbol{X}$ and β?
  - What is the difference between a training data set and a test data set?
  - What is gradient descent?
  - What is a quantizer?