DATA2002

Multiple regression prediction and performance

Garth Tarr

THE UNIVERSITY OF SYDNEY

Prediction

Performance

Prediction

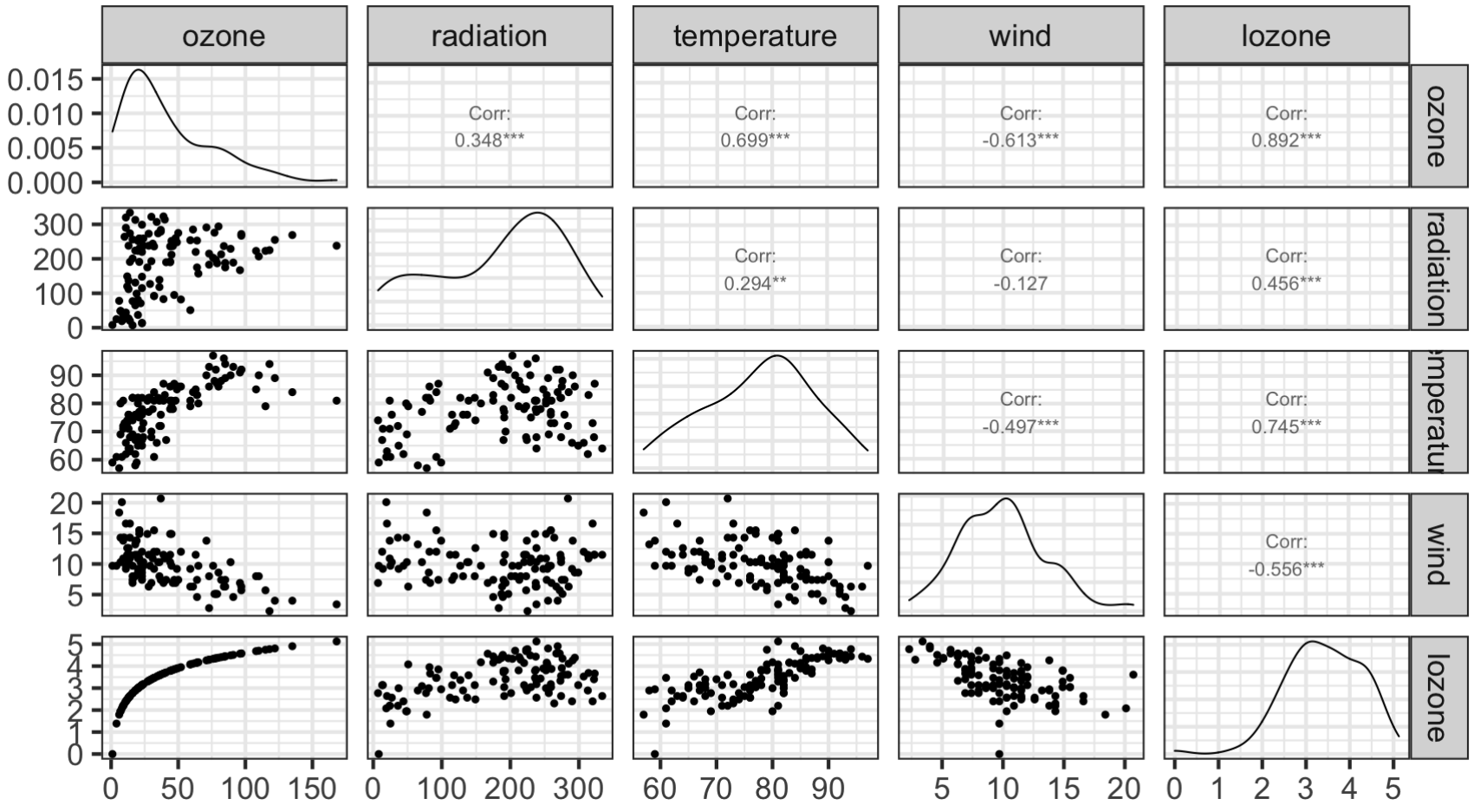
Recall our fitted ozone model

```
library(tidyverse)
data(environmental, package = "lattice")
environmental = environmental %>%
  mutate(lozone = log(ozone))
lm3 = lm(lozone ~ radiation + temperature + wind, environmental)
lm3
```

```
##
## Call:
## lm(formula = lozone ~ radiation + temperature + wind, data = environmental)
##
## Coefficients:
## (Intercept)    radiation  temperature         wind
##   -0.261174     0.002515     0.049163    -0.061593
```

$$\widehat{\log(\text{ozone})} = -0.261174 + 0.002515\,\text{radiation} + 0.049163\,\text{temperature} - 0.061593\,\text{wind}$$

```
library(GGally)
GGally::ggpairs(environmental) + theme_bw(base_size = 22)
```



Prediction

Say we want to predict the (log) ozone when the solar ratiation is 200 langley, the temperature is 90 degrees Fahrenheit and the average wind speed is 15 miles per hour.

We can substitute these into our fitted model:

$$\begin{aligned}\widehat{\log(\text{ozone})} &= -0.261174 + 0.002515\,\text{radiation} + 0.049163\,\text{temperature} - 0.061593\,\text{wind} \\ &= -0.261174 + 0.002515 \times 200 + 0.049163 \times 90 - 0.061593 \times 15 \\ &= 3.74\end{aligned}$$

Prediction in R

We need to generate a new data frame with the same column names as the original variables:

```
new_obs = data.frame(radiation = 200, temperature = 90, wind = 15)
```

And then feed this into the `predict()` function:

```
predict(lm3, new_obs, interval = "prediction", level = 0.90)
```

```
##        fit      lwr      upr
## 1 3.742554 2.867449 4.617659
```

```
predict(lm3, new_obs, interval = "confidence", level = 0.90)
```

```
##        fit      lwr     upr
## 1 3.742554 3.510278 3.97483
```

We have two options for the interval type: **prediction interval** and **confidence interval**.

Two kinds of "prediction"

- Estimate the **average log ozone concentration** when the solar ratiation is 200 langley, the temperature is 90 degrees Fahrenheit and the average wind speed is 15 miles per hour and give a 90% estimation interval of this estimate.
- Predict the **log ozone concentration on a day** when solar ratiation is 200 langley, the temperature is 90 degrees Fahrenheit and the average wind speed is 15 miles per hour. Give a 90% prediction interval of this **prediction**.

Prediction vs confidence intervals

Take a regression model with n observations and p regressors:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

Given a new observation vector $\boldsymbol{x_0}$, the predicted value for that observation would be

$$\mathrm{E}(Y \mid \boldsymbol{x_0}) = \hat{y}_0 = \boldsymbol{x}_0'\hat{\boldsymbol{\beta}}.$$

A consistent estimator of the variance of this prediction is

$$\widehat{\mathrm{Var}}(\hat{y}_0) = \hat{\sigma}^2 \boldsymbol{x}_0'(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{x}_0,$$

where

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N-k} = \frac{\sum_{i=1}^{N} r_i^2}{N-k}.$$

[Think of this as the **mean square residual** in an ANOVA context.]

Prediction vs confidence intervals

The forecast error for **a particular** y_0 is

$$\hat{e} = y_0 - \hat{y}_0 = (\boldsymbol{x}_0'\boldsymbol{\beta} + \varepsilon_0) - \hat{y}_0.$$

Theere is zero covariance between ε_0 and $\hat{\boldsymbol{\beta}}$ so,

$$\operatorname{Var}(\hat{e}) = \operatorname{Var}(\hat{y}_0) + \operatorname{Var}(\varepsilon_0),$$

and a consistent estimator of that is

$$\operatorname{Var}(\hat{e}) = \hat{\sigma}^2 \boldsymbol{x}_0'(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{x}_0 + \hat{\sigma}^2.$$

The $1-\alpha$ **confidence** interval will be: $\hat{y}_0 \pm t^\star \sqrt{\operatorname{Var}(\hat{y}_0)}$, where $\operatorname{Var}(\hat{y}_0) = \hat{\sigma}^2 \boldsymbol{x}_0(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{x}_0'$

The $1-\alpha$ **prediction** interval will be wider: $\hat{y}_0 \pm t^\star \sqrt{\operatorname{Var}(\hat{e})}$

Prediction vs confidence intervals

```
predict(lm3, new_obs, interval = "prediction",
        level = 0.90, se.fit = TRUE)
```

```
## $fit
##        fit      lwr      upr
## 1 3.742554 2.867449 4.617659
##
## $se.fit
## [1] 0.139991
##
## $df
## [1] 107
##
## $residual.scale
## [1] 0.5085019
```

```
predict(lm3, new_obs, interval = "confidence",
        level = 0.90)
```

```
##        fit      lwr     upr
## 1 3.742554 3.510278 3.97483
```

Quantile for 90% intervals:

```
qt(0.95, 107)
```

```
## [1] 1.659219
```

Prediction interval $\hat{y}_0 \pm t^\star \sqrt{\text{Var}(\hat{e})}$

$$3.74 \pm 1.659 \times \sqrt{0.14^2 + 0.51^2}$$

Confidence interval $\hat{y}_0 \pm t^\star \sqrt{\text{Var}(\hat{y}_0)}$

$$3.74 \pm 1.659 \times 0.14$$

Effect of variance on intervals

Our population model is:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

where $\boldsymbol{\varepsilon} \sim N_n(0, \sigma^2)$.

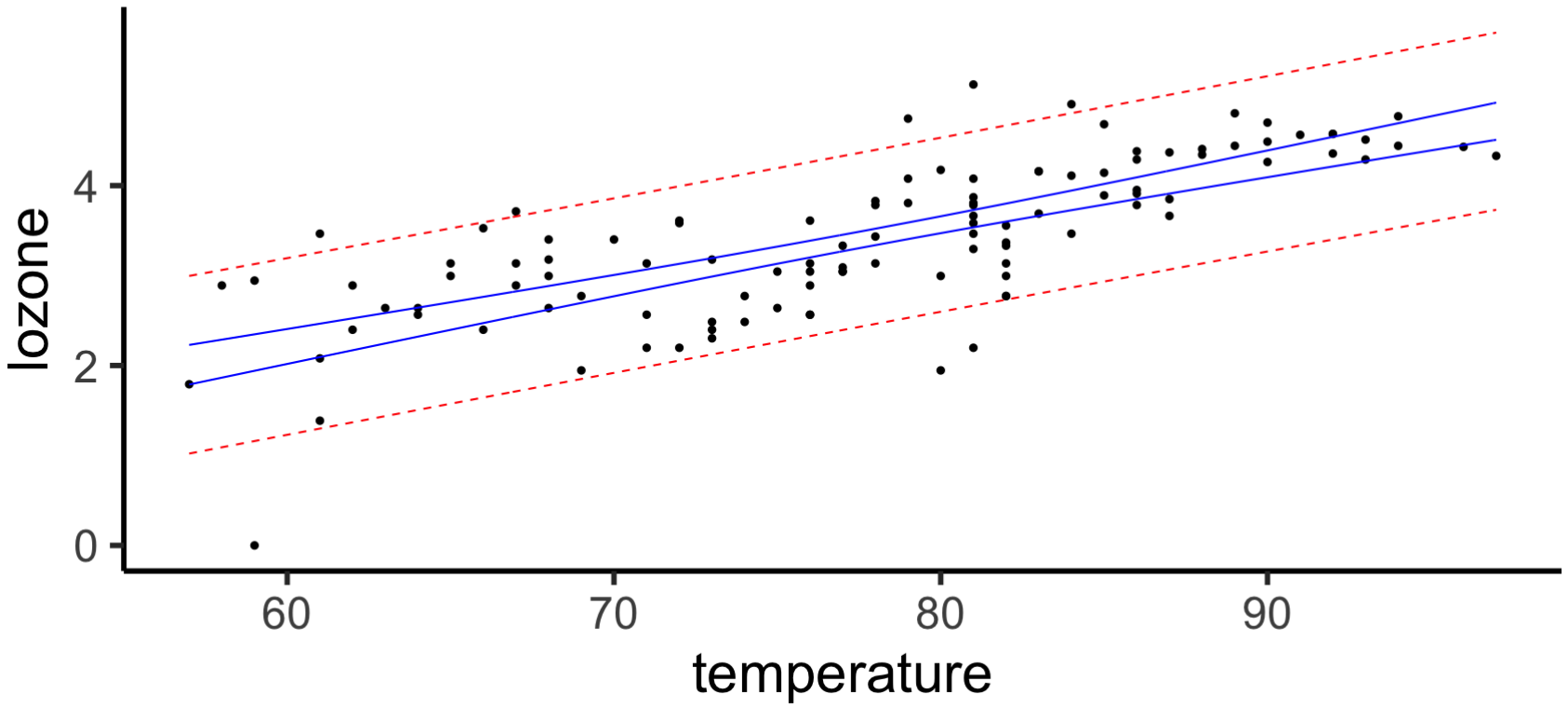
1. The smaller the σ^2, the better the fit and hence the smaller the variances for $\hat{\boldsymbol{\beta}}$ and $\hat{y}_0$.
2. The larger the spread of our x variables, the more information we have about how Y responds to each x variable and hence the smaller the variances for $\hat{\boldsymbol{\beta}}$ and $\hat{y}_0$.
3. The larger the sample size n, the smaller the variances for $\hat{\boldsymbol{\beta}}$ and $\hat{y}_0$.
4. The closer is $\boldsymbol{x}_0$ to $\bar{\boldsymbol{x}}$ (the componentwise mean of the design matrix), the smaller the variances of $\hat{y}_0$

Confidence and prediction intervals

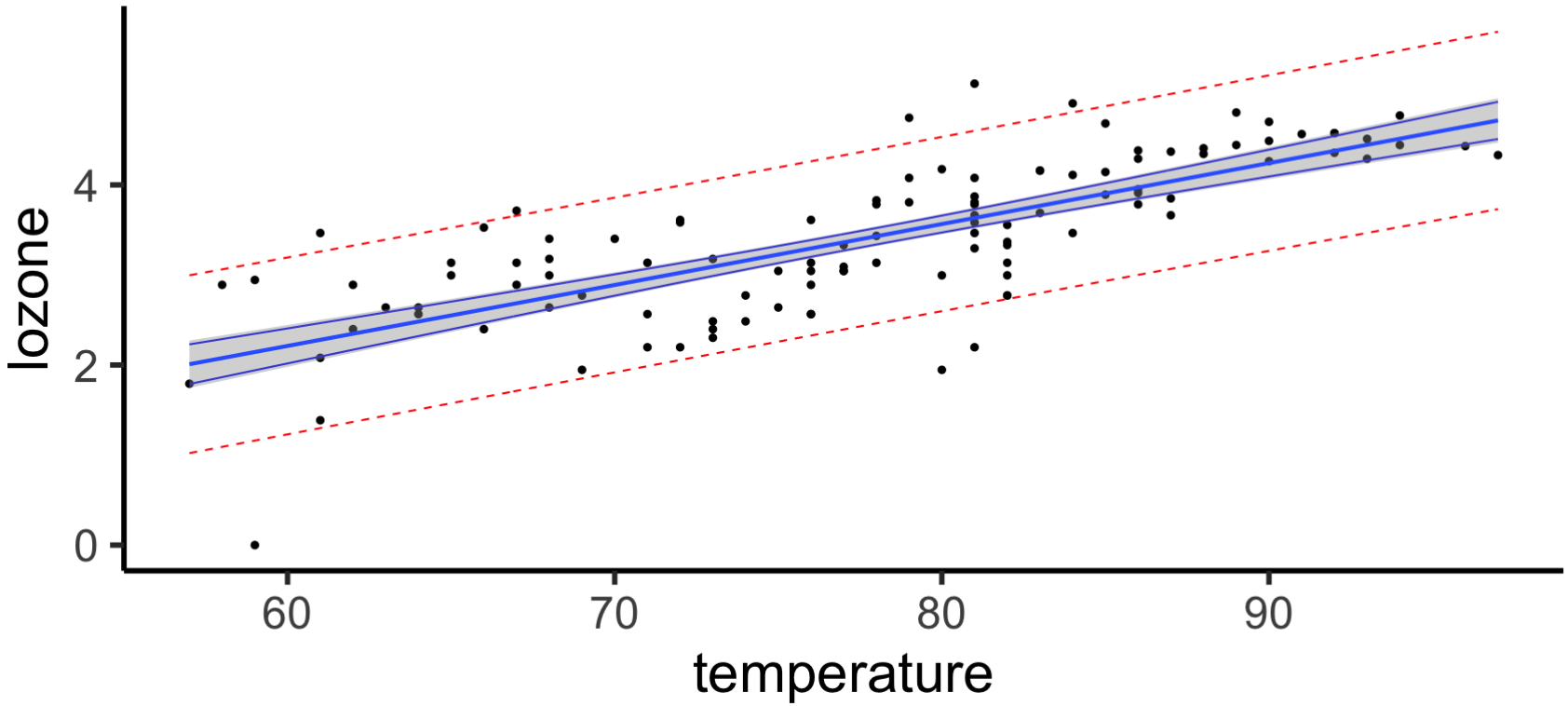
For illustration purposes, let's return to the **simple linear regression** of log ozone on temperature.

```
lm2 = lm(lozone ~ temperature, data = environmental)
new_temp = data.frame(temperature = seq(from = min(environmental$temperature),
                                        to = max(environmental$temperature),
                                        by = 0.1))
pi = predict(lm2, new_temp, interval = "prediction", level = 0.90)
ci = predict(lm2, new_temp, interval = "confidence", level = 0.90)
interval_df = data.frame(
  pi_upper = pi[,"upr"],
  pi_lower = pi[,"lwr"],
  ci_upper = ci[,"upr"],
  ci_lower = ci[,"lwr"],
  temperature = new_temp$temperature
)
```

```
ggplot(environmental, aes(x = temperature, y = lozone)) +
  geom_point() + theme_classic(base_size = 30) +
  geom_line(data = interval_df, aes(y=pi_lower), color = "red", linetype = 2) +
  geom_line(data = interval_df, aes(y=pi_upper), color = "red", linetype = 2) +
  geom_line(data = interval_df, aes(y=ci_lower), color = "blue", linetype = 1) +
  geom_line(data = interval_df, aes(y=ci_upper), color = "blue", linetype = 1)
```



```
ggplot(environmental, aes(x = temperature, y = lozone)) +
  geom_point() + theme_classic(base_size = 30) +
  geom_line(data = interval_df, aes(y=pi_lower), color = "red", linetype = 2) +
  geom_line(data = interval_df, aes(y=pi_upper), color = "red", linetype = 2) +
  geom_line(data = interval_df, aes(y=ci_lower), color = "blue", linetype = 1) +
  geom_line(data = interval_df, aes(y=ci_upper), color = "blue", linetype = 1) +
  geom_smooth(method = "lm", se = TRUE)
```



Performance

In sample performance vs out of sample performance

In sample

- E.g. r^2 comparing the simple linear regression to the full model

```
summary(lm2)$r.squared # lozone ~ temperature
```

```
## [1] 0.5547615
```

```
summary(lm3)$r.squared # lozone ~ radiation + temperature + wind
```

```
## [1] 0.664515
```

- Doesn't protect against **over fitting**

Out of sample

- How well do we predict observations that we didn't use to build the model?

Comparing simple linear regression with the full model

Out of sample performance

We could think about building a **training** set and using it to predict observations from a **test** set.

```
n = nrow(environmental)
n
```

```
## [1] 111
```

```
n_train = floor(0.8*n)
n_test = n - n_train
grp_labs = rep(c("Train","Test"), times = c(n_train, n_test))
environmental$grp = sample(grp_labs)
train_dat = environmental %>% filter(grp == "Train")
lm_simple_train = lm(lozone ~ temperature, data = train_dat)
lm_full_train = lm(lozone ~ radiation + temperature + wind, data = train_dat)
test_dat = environmental %>% filter(grp == "Test")
simple_pred = predict(lm_simple_train, newdata = test_dat)
full_pred = predict(lm_full_train, newdata = test_dat)
```

Comparing simple linear regression with the full model

Root mean square error

How can we compare the predictions from the two models? Compare them to the observed values using the root mean square error:

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

```
simple_mse = mean((test_dat$lozone - simple_pred)^2)
sqrt(simple_mse)
```

```
## [1] 0.4370441
```

```
full_mse = mean((test_dat$lozone - full_pred)^2)
sqrt(full_mse)
```

```
## [1] 0.4717252
```

Comparing simple linear regression with the full model

Mean absolute error

An alternative measure of performance, less influenced by outliers is the **mean absolute error**,

$$\text{MAE} = \frac{\sum_{i=1}^{m} |y_i - \hat{y}_i|}{m}$$

```
simple_mae = mean(abs(test_dat$lozone - simple_pred))
simple_mae
```

```
## [1] 0.3373928
```

```
full_mae = mean(abs(test_dat$lozone - full_pred))
full_mae
```

```
## [1] 0.3807022
```

Out of sample performance

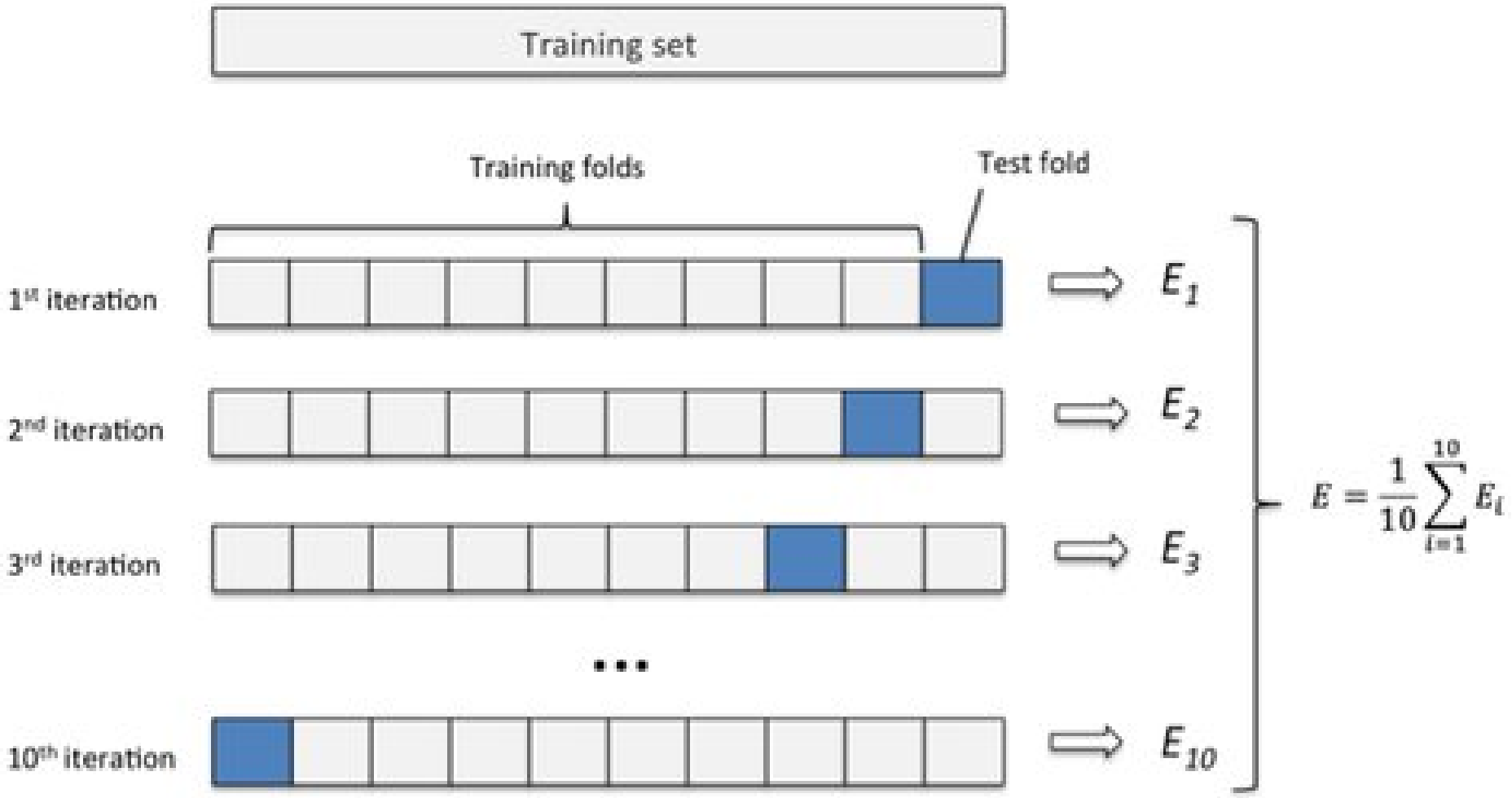
k-fold cross-validation (CV) estimation

- Data randomly divided into k subsets of (nearly) equal size
- Estimate your model by leaving one subset out
- Use your estimated model to predict the observations left out
- Compute error rates on the left out set
- Repeat k times (for each of the subsets)
- Average the error rate over the k runs

Bias-variance tradeoff: smaller k can give larger bias but smaller variance

Computationally intensive.

10-fold cross validation



10-fold cross validation

Step 1: divide our data up into 10 folds there are 111 observations, so we have 9 folds of 11 observations and 1 fold of 12 observations.

```
set.seed(2)
nrow(environmental)
```

```
## [1] 111
```

```
environmental$grp = NULL # remove the grp variable we added previously
fold_id = c(1, rep(1:10, each = 11))
environmental$fold_id = sample(fold_id, replace = FALSE)
head(environmental)
```

```
##   ozone radiation temperature wind   lozone fold_id
## 1    41       190          67  7.4 3.713572       8
## 2    36       118          72  8.0 3.583519       8
## 3    12       149          74 12.6 2.484907       7
## 4    18       313          62 11.5 2.890372       1
## 5    23       299          65  8.6 3.135494       3
## 6    19        99          59 13.8 2.944439       1
```

10-fold cross validation

Step 2: estimate the model leaving one fold out, make predictions on the test set and calculate the error rate

```
k = 10
simple_mse = full_mse = vector(mode = "numeric", length = k)
simple_mae = full_mae = vector(mode = "numeric", length = k)
for(i in 1:k) {
  test_set = environmental[fold_id == i,]
  training_set = environmental[fold_id != i,]
  simple_lm = lm(lozone ~ temperature, data = training_set)
  simple_pred = predict(simple_lm, test_set)
  simple_mse[i] = mean((test_set$lozone - simple_pred)^2)
  simple_mae[i] = mean(abs(test_set$lozone - simple_pred))
  full_lm = lm(lozone ~ radiation + temperature + wind, data = training_set)
  full_pred = predict(full_lm, test_set)
  full_mse[i] = mean((test_set$lozone - full_pred)^2)
  full_mae[i] = mean(abs(test_set$lozone - full_pred))
}
```

10-fold cross validation

Step 3: aggregate the errors over the 10 folds

```
cv_res = tibble(simple_mse, full_mse,
                simple_mae, full_mae)
cv_res
```

```
## # A tibble: 10 × 4
##    simple_mse full_mse simple_mae full_mae
##         <dbl>    <dbl>      <dbl>    <dbl>
##  1     0.437    0.400       0.527    0.552
##  2     0.955    0.839       0.784    0.722
##  3     0.307    0.244       0.489    0.418
##  4     0.348    0.194       0.396    0.297
##  5     0.176    0.134       0.360    0.313
##  6     0.376    0.190       0.471    0.372
##  7     0.389    0.260       0.486    0.395
##  8     0.0879   0.0783      0.249    0.206
##  9     0.160    0.0975      0.322    0.284
## 10     0.252    0.251       0.414    0.397
```

Root mean square errors:

```
c(sqrt(mean(simple_mse)),
  sqrt(mean(full_mse))) %>% round(2)
```

```
## [1] 0.59 0.52
```

Mean absolute errors:

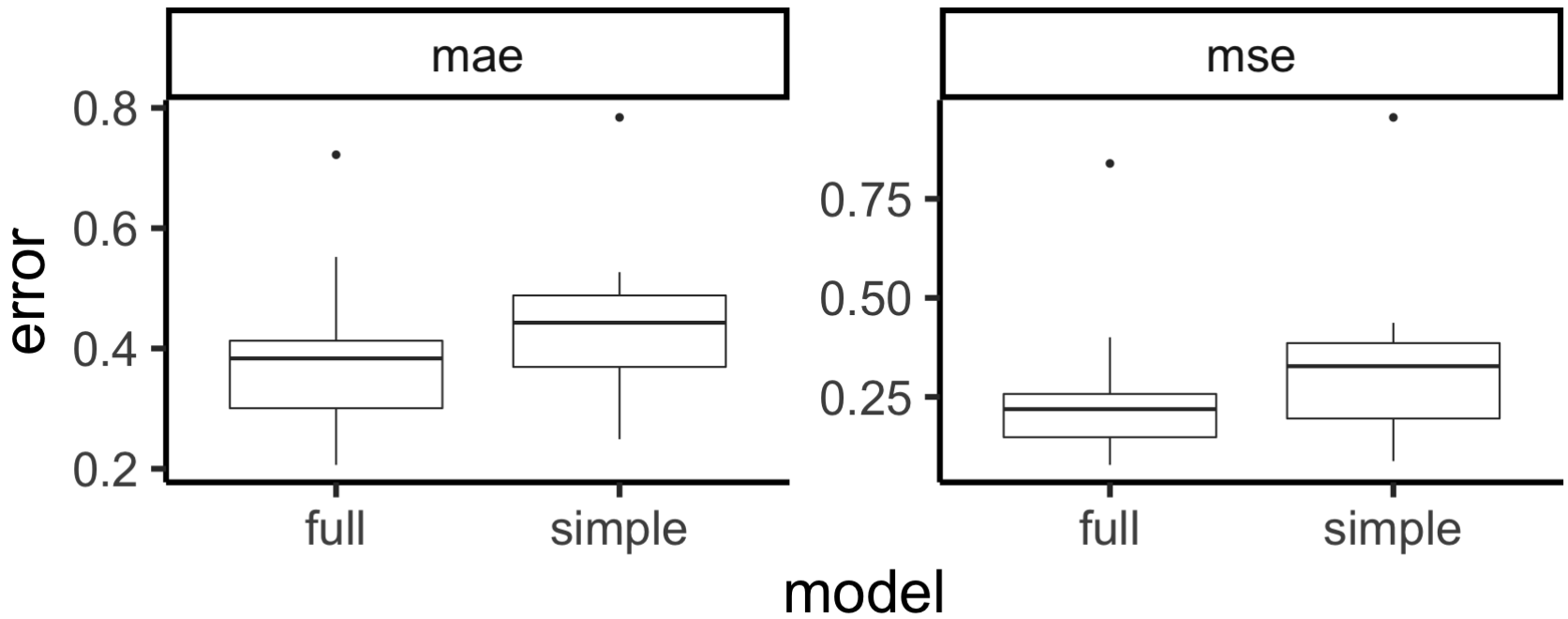
```
c(mean(simple_mae),
  mean(full_mae)) %>% round(2)
```

```
## [1] 0.45 0.40
```

10-fold cross validation

We could visualise the error rates for each of the 10 folds:

```
cv_res %>% gather(key = "metric", value = "error") %>%
  separate(col = metric, into = c("model","metric")) %>%
  ggplot(aes(x = model, y = error)) + facet_wrap(~metric, scales = "free_y") +
  geom_boxplot()
```



The `caret` package (Classification And REgression Training)

```
library(caret)
cv_full = train(
  lozone ~ radiation + temperature + wind, environmental,
  method = "lm",
  trControl = trainControl(
    method = "cv", number = 10,
    verboseIter = FALSE
  )
)
cv_full
```

```
## Linear Regression
##
## 111 samples
##   3 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 100, 100, 99, 100, 101, 99, ...
## Resampling results:
##
##   RMSE       Rsquared  MAE
##   0.5110818  0.697293  0.3990078
```

```
cv_simple = train(
  lozone ~ temperature,
  environmental,
  method = "lm",
  trControl = trainControl(
    method = "cv", number = 10,
    verboseIter = FALSE
  )
)
cv_simple
```

```
## Linear Regression
##
## 111 samples
##   1 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 99, 99, 100,
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.5639179  0.5597842  0.4403246
```

References

Baumer, Kaplan, and Horton (2017) Appendix E Regression modeling

Baumer, B. S., D. T. Kaplan, and N. J. Horton (2017). *Modern Data Science with R*. Boca Raton: Chapman and Hall/CRC. URL: https://mdsr-book.github.io/index.html.

Jed Wing, M. K. C. from, S. Weston, A. Williams, C. Keefer, A. Engelhardt, T. Cooper, Z. Mayer, B. Kenkel, the R Core Team, M. Benesty, et al. (2018). *caret: Classification and Regression Training*. R package version 6.0-80. URL: https://CRAN.R-project.org/package=caret.