Lecture 6: Hypothesis testing and Linear Regression

October 19, 2017

Contents

Useful Books

- “An introduction to Statistical Learning” [ISL] by James, Witten, Hastie and Tibshirani
- “Elements of statistical learning” [ESL] by Hastie, Tibshirani and Friedman
- “Introduction to Linear Regression Analysis” by Montgomery, Peck, Vinning

Hypothesis testing

Hypothesis testing can answer questions:

- **Is the measured quantity equal to/higher/lower than a given threshold?** e.g. is the number of faulty items in an order statistically higher than the one guaranteed by a manufacturer?
- **Is there a difference between two groups or observations**? e.g. Do treated patient have a higher survival rate than the untreated ones?
- **Is the level of one quantity related to the value of the other quantity?** e.g. Is hyperactivity related to eating sugar? Is lung cancer related to smoking?

Everyday life examples (top answer from Quora):

- Test weather route A or route B is the faster way to get from your home to your school.
- Test whether acetaminophen (Tylenol) or ibuprofen (Advil) helps faster with your headaches.
- If you are 21 or older, test whether Tequila or beer gives you worse hangover.
- Test if you run faster in the morning compared to the afternoon.
- Test if you weight is lower in the morning compared to the afternoon.

To perform a hypothesis test you need to:

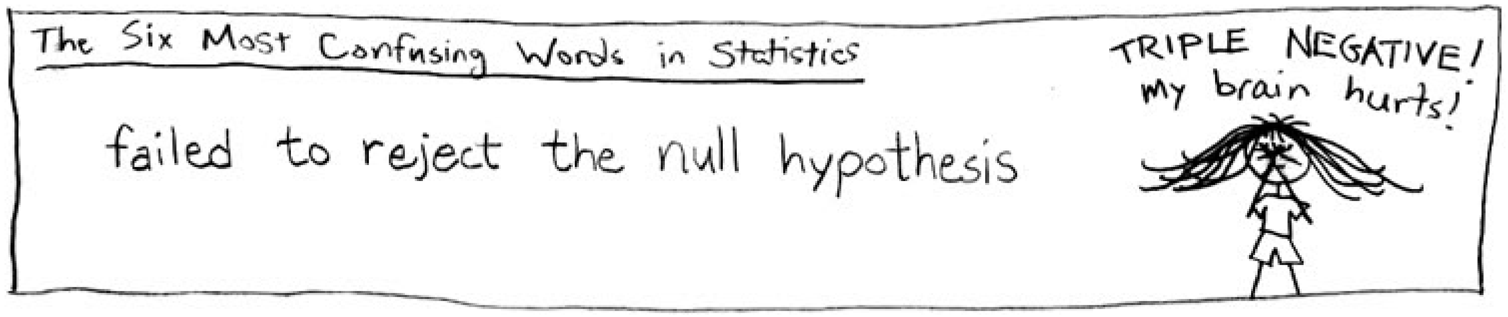
1. Define the null and alternative hypotheses.
2. Choose level of significance α.
3. Pick and compute test statistics.
4. Compute the p-value.
5. Check whether to reject the null hypothesis by comparing p-value to α.
6. Draw conclusion from the test.

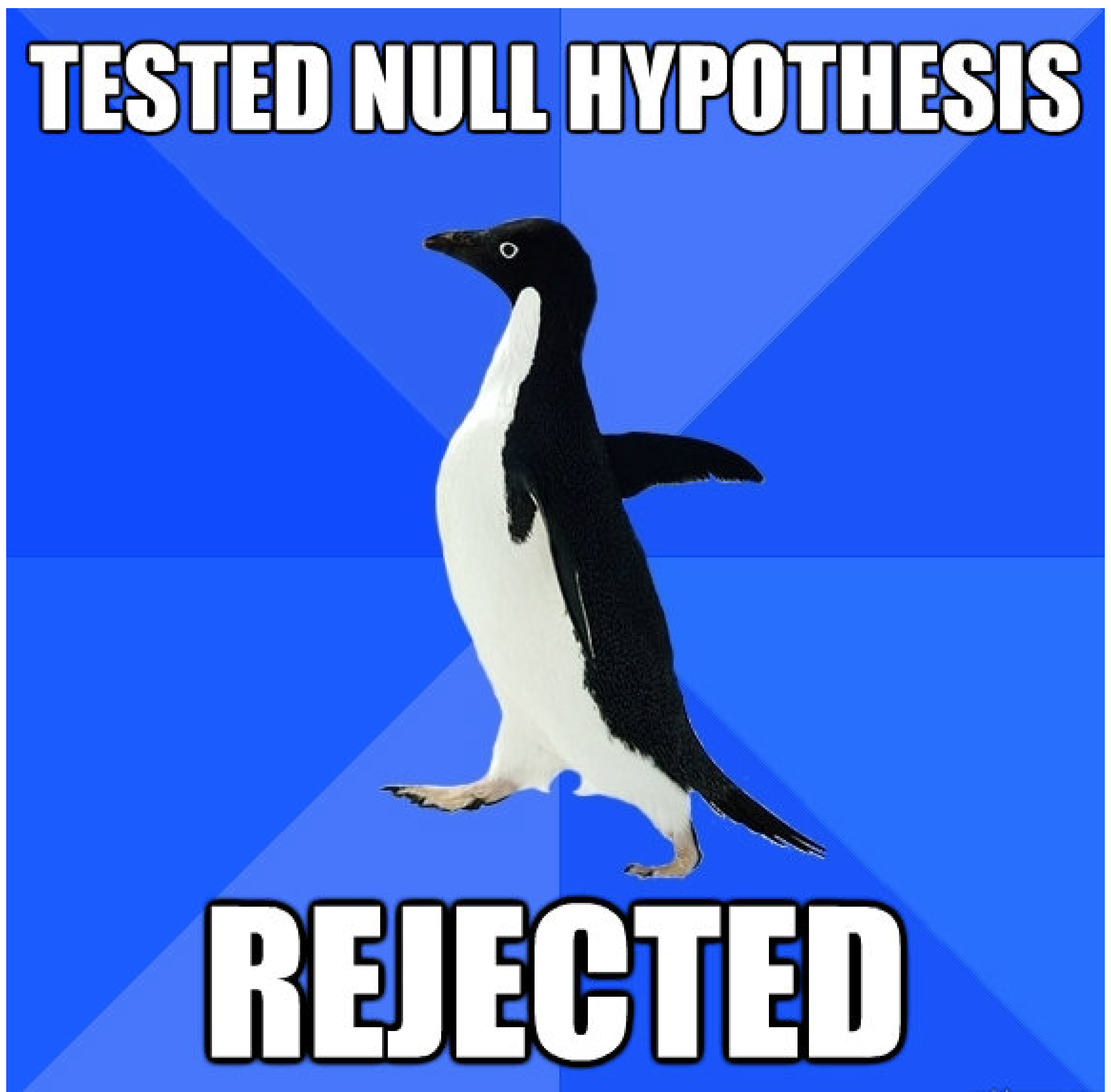
Null and alternative hypotheses

The null hypothesis (H_0): A statement assumed to be true unless it can be shov to be incorrect beyond a reasonable doubt. This is something one usually atter to disprove or discredit.

The alternate hypothesis (H_1): A claim that is contradictory to H0 and what w conclude when we reject H0.

H0 and H1 are on purpose set up to be contradictory, so that one **can collect a examine data to decide if there is enough evidence to reject the null hypothe: or not**.


The Six Most Confusing Words in Statistics
failed to reject the null hypothesis
TRIPLE NEGATIVE! my brain hurts!


TESTED NULL HYPOTHESIS
REJECTED
quickmeme.com

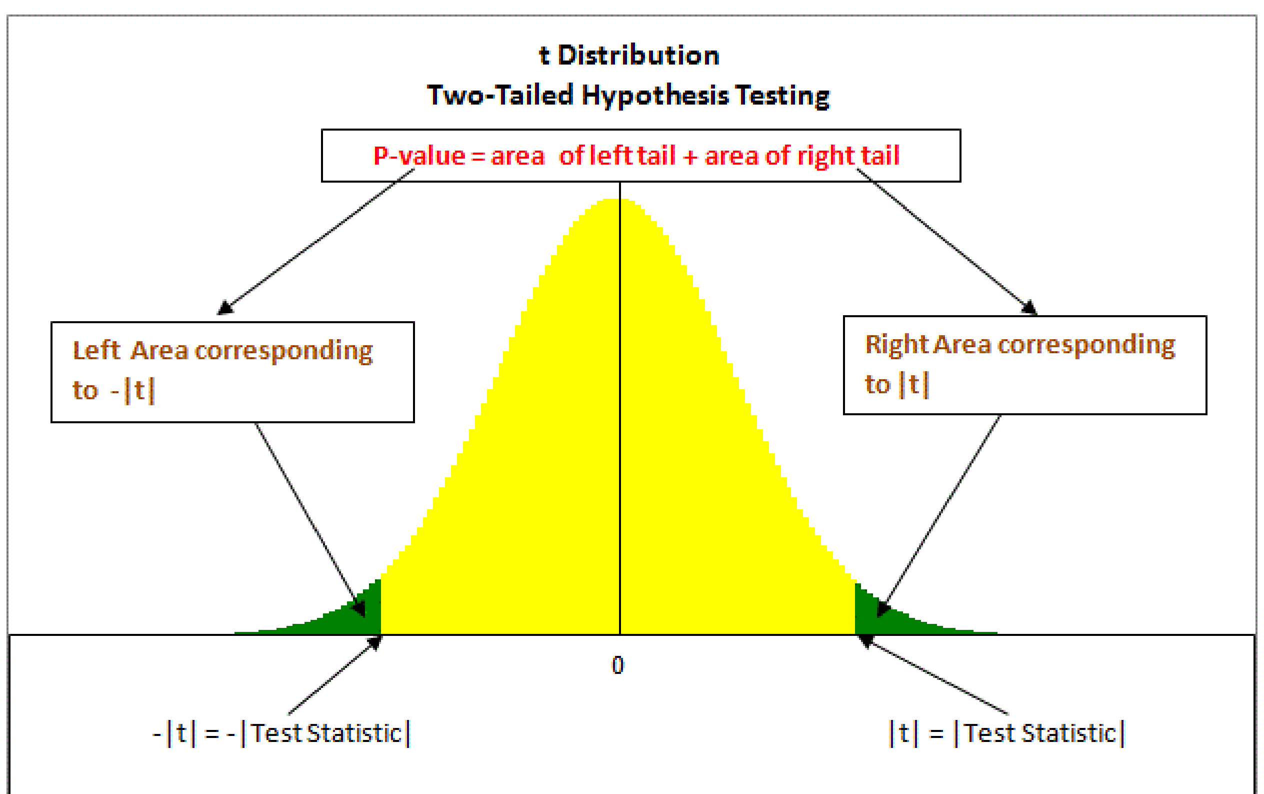
Student's t-test

- William Gosset (1908), a chemist at **the Guiness brewery**.
- Published in Biometrika under a **pseudonym Student**.
- Used to select best yielding varieties of barley.
- Now one of the standard/traditional methods for hypothesis testing.

Distribution of the statistic

p-value is the probability of an observed (or more extreme) result assuming tha
the null hypothesis is true, i.e. $P[observations \mid H_0]$. Note that:

$$P[observations \quad | \quad hypothesis] \neq P[hypothesis \quad | \quad ovservations]$$



This is the reason why p-values sh NOT be used a "ranking"/"sco system for your hypotheses. You sh only use it to potentially reject you hypothesis. Null hypothesis cann proven true, **you can only fail to reje**

p-value

- p-value is the **probability of obtaining a result equal to or “more extreme” than what was actually observed, when the null hypothesis is true**.
- A small p-value (typically ≤ 0.05) indicates strong evidence against the null hypothesis, so you reject the null hypothesis.
- A large p-value (> 0.05) indicates weak evidence against the null hypothesis, so you do NOT to reject the null hypothesis.

Dataset

- A built-in dataset, `mtcars`, that comes from a 1974 issue of Motor Trends magazine.

```
data("mtcars")
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

- rows correspond to car models,
- column are car attributes: miles per gallon, number of cylinders, displacement, transmission etc.

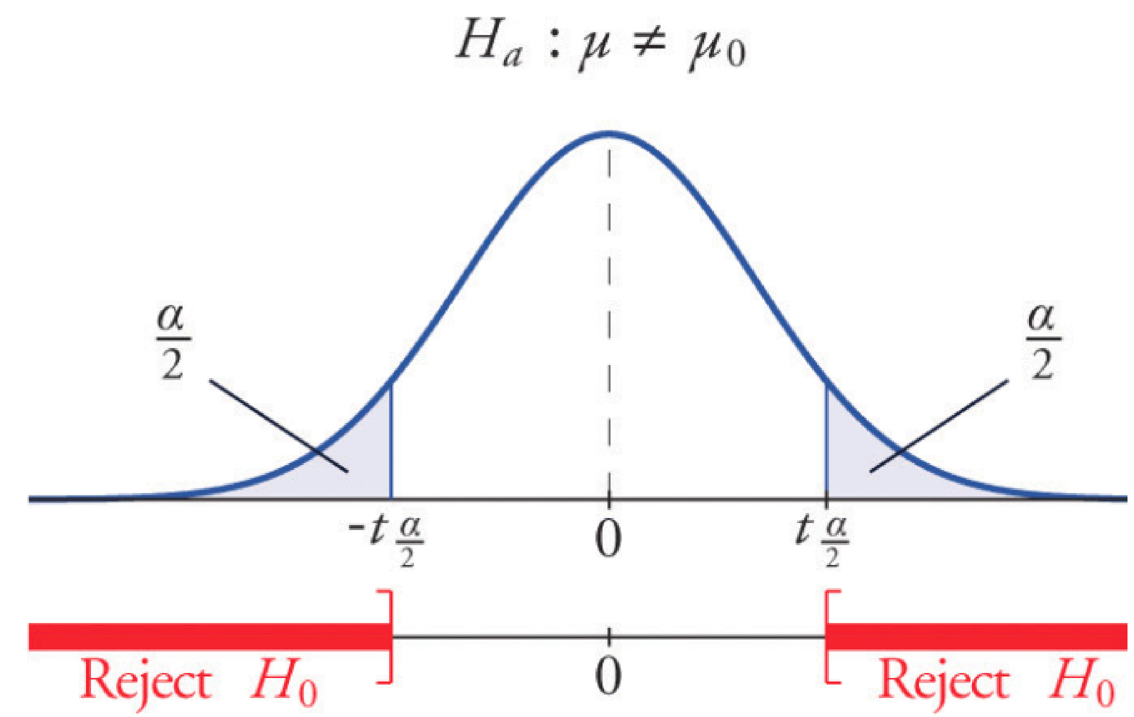
Testing mpg equal to a value

Is the mean fuel efficiency (mpg) in the cars in `mtcars` statistically equal to 25?

Test the null hypothesis:

$$H_0 : \mu = 25$$
$$H_a : \mu \neq 25$$

where μ is the mean mpg of cars in the dataset



```r
mtcars$mpg
```

```
##  [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2
## [15] 10.4 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4
## [29] 15.8 19.7 15.0 21.4
```

```r
tt <- t.test(x = mtcars$mpg, mu = 25, alternative = "two.sided")
tt
```

```
## 
##  One Sample t-test
## 
## data:  mtcars$mpg
## t = -4.6079, df = 31, p-value = 6.587e-05
## alternative hypothesis: true mean is not equal to 25
## 95 percent confidence interval:
##  17.91768 22.26357
## sample estimates:
## mean of x 
##  20.09062
```

```r
names(tt)
```

```
## [1] "statistic"   "parameter"   "p.value"     "conf.int"    "estimate"
## [6] "null.value"  "alternative" "method"      "data.name"
```

```r
# The p-value:
tt$p.value
```

```
## [1] 6.586738e-05
```

```r
# The 95% confidence interval for the mean:
tt$conf.int
```

```
## [1] 17.91768 22.26357
## attr(,"conf.level")
## [1] 0.95
```

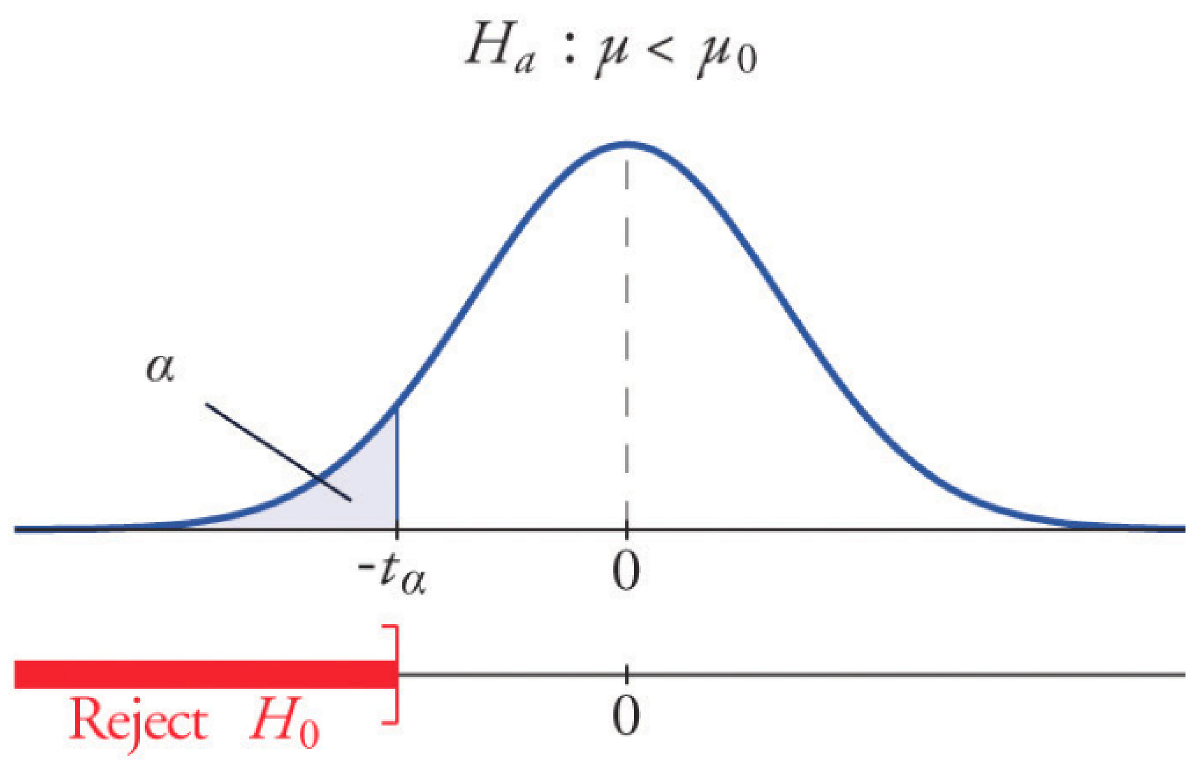
Testing mpg smaller than a value

Is the mean fuel efficiency (mpg) in the cars in `mtcars` statistically less than 25?

Test the null hypothesis:

$$H_0 : \mu = 25$$
$$H_a : \mu < 25$$

where μ is the mean mpg of cars in the dataset



```r
tt <- t.test(x = mtcars$mpg, mu = 25, alternative = "less")
tt
```

```
##
##  One Sample t-test
##
## data:  mtcars$mpg
## t = -4.6079, df = 31, p-value = 3.293e-05
## alternative hypothesis: true mean is less than 25
## 95 percent confidence interval:
##      -Inf 21.89707
## sample estimates:
## mean of x
##  20.09062
```

Testing difference between groups

Is the fuel efficiency (mpg) the same for the cars with automatic and manual transmission?

Test the null hypothesis:

$$H_0 : \mu_a = \mu_m$$
$$H_a : \mu_a \neq \mu_m$$

where μ_a mean mpg of automatic cars and μ_m is the mean mpg of manual cars.

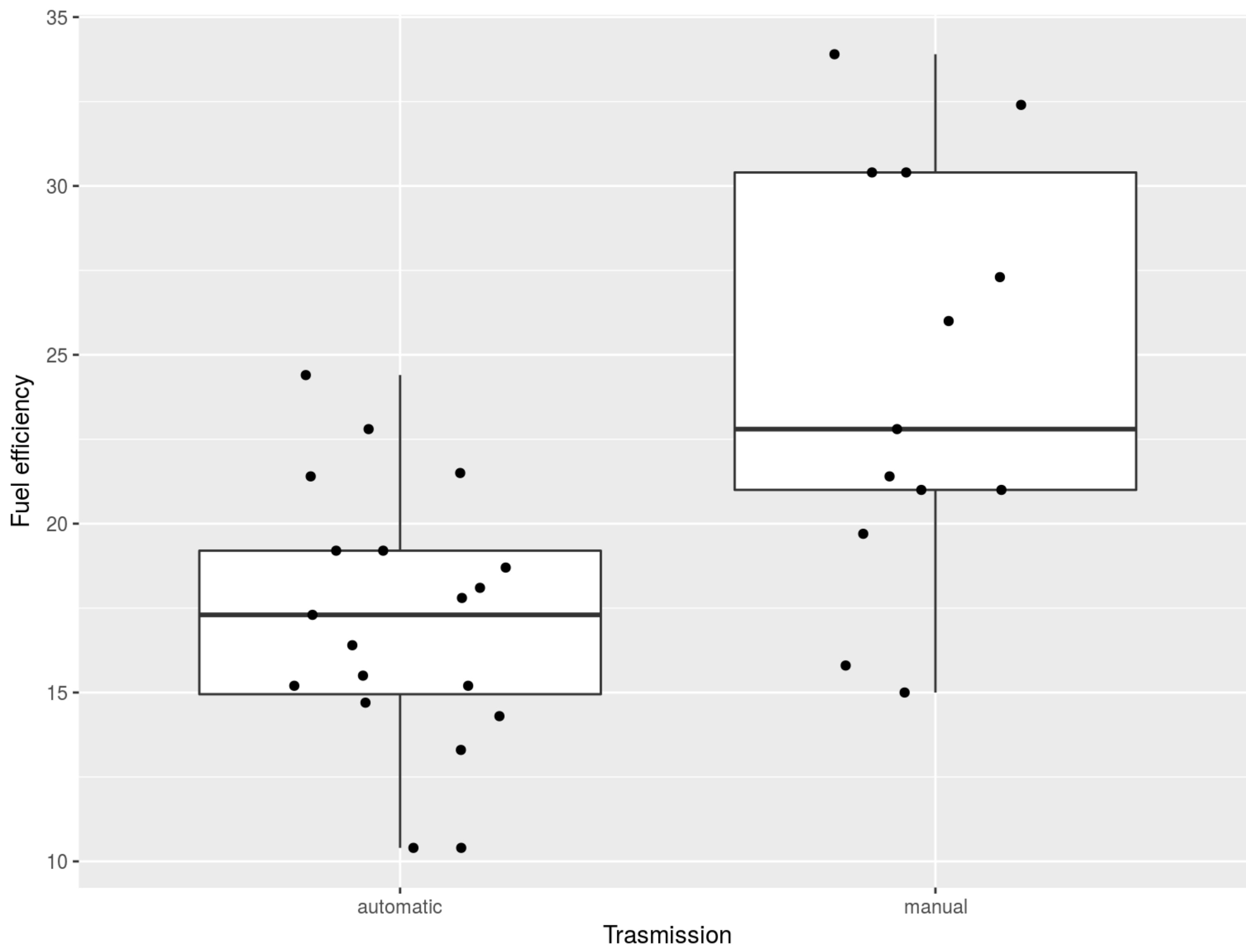
Convert the column `am` (transmission) to a factor:

```r
#  convert am (0 = automatic, 1 = manual) column to a factor
mtcars <- mtcars %>%
  mutate(am = factor(am, levels = c(0, 1),
                              labels = c("automatic", "manual")))
head(mtcars)
```

```
##    mpg cyl disp  hp drat    wt  qsec vs        am gear carb
## 1 21.0   6  160 110 3.90 2.620 16.46  0    manual    4    4
## 2 21.0   6  160 110 3.90 2.875 17.02  0    manual    4    4
## 3 22.8   4  108  93 3.85 2.320 18.61  1    manual    4    1
## 4 21.4   6  258 110 3.08 3.215 19.44  1 automatic    3    1
## 5 18.7   8  360 175 3.15 3.440 17.02  0 automatic    3    2
## 6 18.1   6  225 105 2.76 3.460 20.22  1 automatic    3    1
```

First, visualize the data

```
library(ggplot2)
ggplot(mtcars, aes(x = am, y = mpg)) + geom_boxplot() +
  xlab("Trasmission") + ylab("Fuel efficiency") +
  geom_jitter(width = 0.2, height = 0)
```



The R implementation of the Student's t-test is `t.test()` function:

```
tt <- t.test(x = mtcars$mpg[mtcars$am == "automatic"],
             y = mtcars$mpg[mtcars$am == "manual"])
# or
(tt <- t.test(formula = mpg ~ am, data = mtcars))
```

```
##
##  Welch Two Sample t-test
##
## data:  mpg by am
## t = -3.7671, df = 18.332, p-value = 0.001374
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -11.280194  -3.209684
## sample estimates:
## mean in group automatic    mean in group manual
##                17.14737                24.39231
```

- **A tilde symbol, ~**, means "explained by".

Exercise

- Go to the “Lec6_Exercises.Rmd” file, which can be downloaded from the class website under the Lecture tab.
- Complete Exercise 1.

Linear Regression

Linear Regression

- Regression is a supervised learning method, whose goal is inferring the relationship between input data, x, and a **continuous** response variable, y.
- **Linear regression** is a type of regression where **y is modeled as a linear function of x**.
- **Simple linear regression** predicts the output y from a single predictor x.

$$y = \beta_0 + \beta_1 x + \epsilon$$

- **Multiple linear regression** assumes y relies on many covariates:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon = \vec{\beta}\vec{x} + \epsilon$$

- here ϵ denotes a random noise term with zero mean.

Objective function

Linear regression seeks a solution $\hat{y} = \hat{\beta} \cdot \vec{x}$ that **minimizes the difference between the true outcome $\boldsymbol{y}$ and the prediction $\hat{\boldsymbol{y}}$**, in terms of the residual sum squares (RSS).

$$arg \min_{\hat{\beta}} \sum_{i} \left(y^{(i)} - \hat{\beta} x^{(i)}\right)^2$$

Simple Linear Regression

- Predict the mileage per gallon using the weight of the car.
- In R the linear models can be fit with a `lm()` function.

```
fit <- lm(mpg ~ wt, mtcars)
fit
```

```
##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Coefficients:
## (Intercept)           wt
##      37.285       -5.344
```

- Same '~' formula notation as for the `t.test` function.

We can check the details on the fitted model by calling:

```r
summary(fit)
```

```
##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5432 -2.3647 -0.1252  1.4096  6.8727
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
## wt           -5.3445     0.5591  -9.559 1.29e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.046 on 30 degrees of freedom
## Multiple R-squared:  0.7528, Adjusted R-squared:  0.7446
## F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

The coefficients (β) of the model:

```r
(beta <-  coef(summary(fit)))
```

```
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 37.285126   1.877627 19.857575 8.241799e-19
## wt          -5.344472   0.559101 -9.559044 1.293959e-10
```

$\hat{y}$ = predicted `mpg` values for existing observations (cars):

```r
predict(fit)[1:15]
```

```
##        1        2        3        4        5        6        7        8
## 23.28261 21.91977 24.88595 20.10265 18.90014 18.79325 18.20536 20.23626
##        9       10       11       12       13       14       15
## 20.45004 18.90014 18.90014 15.53313 17.35025 17.08302  9.22665
```

To predict the `mpg` for **new observations**, e.g. a new car with weight `wt = 3.1` we can do a manual computation using estimated coefficients:

```
beta[, 1]
```

```
## (Intercept)          wt
##   37.285126   -5.344472
```

```
# beta0 + beta1 * wt
beta[1, 1] + beta[2, 1]* 3.14
```

```
## [1] 20.50349
```

Predictions can be more efficiently computed using `predict()` function:

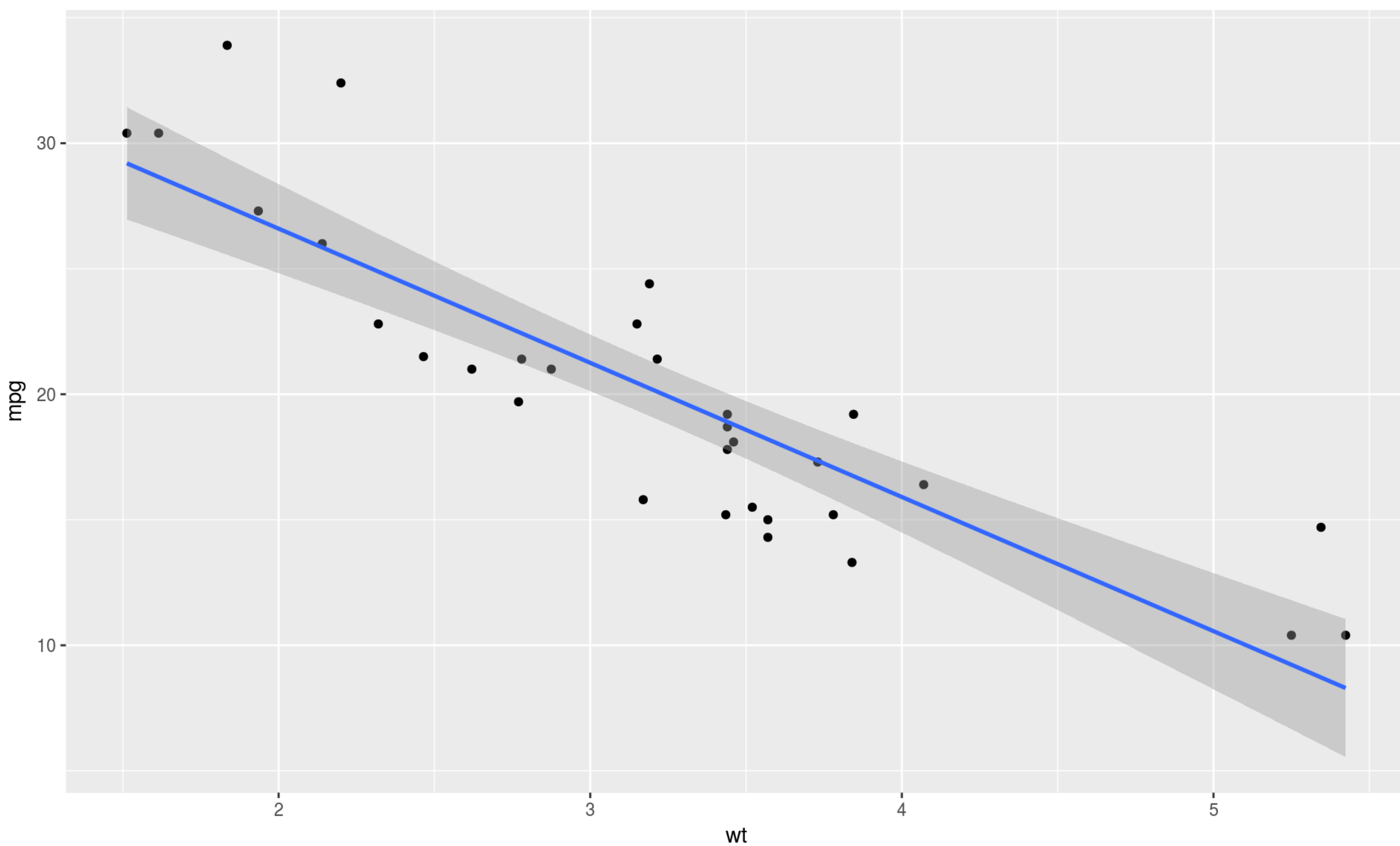
```
# create a data frame with new weights:
newcars <- data.frame(wt = c(2, 2.1, 3.14, 4.1, 4.3))
predict(fit, newcars)
```

```
##        1        2        3        4        5
## 26.59618 26.06174 20.50349 15.37279 14.30390
```

```
# Note the same prediction for `wt = 3.14` as computed previously.
```

`ggplot2` with `geom_smooth()` can plot the data and the fitted `lm` model

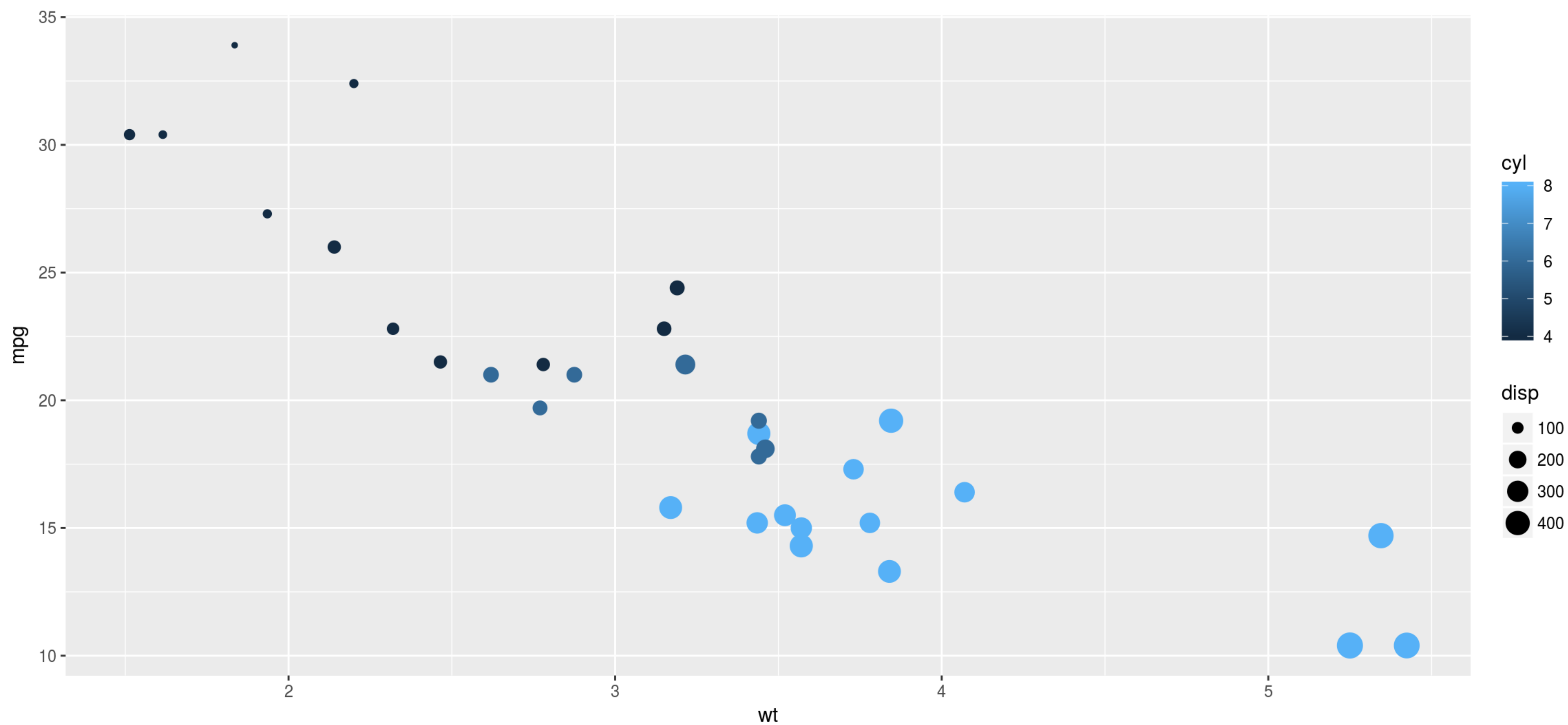
```
ggplot(mtcars, aes(wt, mpg)) + geom_point() + geom_smooth(method="lm")
```



Multiple Linear Regression

We might like to predict `mpg` using weight, displacement and the number of cylinders in the car.

```
ggplot(mtcars, aes(x=wt, y=mpg, col=cyl, size=disp)) + geom_point()
```



```r
mfit <- lm(mpg ~ wt + disp + cyl, data = mtcars)

# Summarize the results
summary(mfit)
```

```
##
## Call:
## lm(formula = mpg ~ wt + disp + cyl, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.4035 -1.4028 -0.4955  1.3387  6.0722
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 41.107678   2.842426  14.462 1.62e-14 ***
## wt          -3.635677   1.040138  -3.495  0.00160 **
## disp         0.007473   0.011845   0.631  0.53322
## cyl         -1.784944   0.607110  -2.940  0.00651 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.595 on 28 degrees of freedom
## Multiple R-squared:  0.8326, Adjusted R-squared:  0.8147
## F-statistic: 46.42 on 3 and 28 DF,  p-value: 5.399e-11
```

To **predict `mpg` for new cars**, you must first create a data frame describing the attributes of the new cars:

```
(newcars <- data.frame(wt = c(2, 2.1, 3.14, 4.1, 4.3),
                       disp = c(100, 200, 500,300, 210),
                       cyl = c(6,6,4,6,8)))
```

```
##     wt disp cyl
## 1 2.00  100   6
## 2 2.10  200   6
## 3 3.14  500   4
## 4 4.10  300   6
## 5 4.30  210   8
```

Then you can compute the predicted `mpg`

```
predict(mfit, newcars)
```

```
##        1        2        3        4        5
## 23.87395 24.25768 26.28834 17.73362 12.76403
```

Interaction terms

- An interaction occurs when **an independent variable has a different effect on the outcome depending on the values of another independent**. variable.
- For example, one variable, x_1 might have a different effect on y within different categories or groups, given by variable x_2.
- If you are not familiar with the concept of the interaction terms, read this.

Models with **interaction effects** can be specified with '*':

```r
mfit_inter <- lm(mpg ~ am * wt, mtcars)
summary(mfit_inter)
```

```
##
## Call:
## lm(formula = mpg ~ am * wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.6004 -1.5446 -0.5325  0.9012  6.0909
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  31.4161     3.0201  10.402 4.00e-11 ***
## ammanual     14.8784     4.2640   3.489  0.00162 **
## wt           -3.7859     0.7856  -4.819 4.55e-05 ***
## ammanual:wt  -5.2984     1.4447  -3.667  0.00102 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.591 on 28 degrees of freedom
## Multiple R-squared:  0.833,  Adjusted R-squared:  0.8151
## F-statistic: 46.57 on 3 and 28 DF,  p-value: 5.209e-11
```

Note that ’*’ generates the interaction terms:

```
mfit_inter <- lm(mpg ~ am * wt, mtcars)
names(coefficients(mfit_inter))
```

```
## [1] "(Intercept)" "ammanual"    "wt"          "ammanual:wt"
```

You can also specify explicitly which terms you want:

```r
mfit_iter2 <- lm(mpg ~ 1 + am + wt + am:wt, mtcars)
summary(mfit_iter2)
```

```
##
## Call:
## lm(formula = mpg ~ 1 + am + wt + am:wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.6004 -1.5446 -0.5325  0.9012  6.0909
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  31.4161     3.0201  10.402 4.00e-11 ***
## ammanual     14.8784     4.2640   3.489  0.00162 **
## wt           -3.7859     0.7856  -4.819 4.55e-05 ***
## ammanual:wt  -5.2984     1.4447  -3.667  0.00102 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.591 on 28 degrees of freedom
## Multiple R-squared:  0.833,  Adjusted R-squared:  0.8151
## F-statistic: 46.57 on 3 and 28 DF,  p-value: 5.209e-11
```

Exercise

- Go to the “Lec6_Exercises.Rmd” file, which can be downloaded from the class website under the Lecture tab.
- Complete Exercise 2.

Extra: Lasso Regression

Choosing a model

- Modern datasets often have “too” many variables, e.g. predict the risk of a disease from the single nucleotide polymorphisms (SNPs) data.
- **Issue:** $n \ll p$ i.e. no. of predictors is much larger than than the no. of observations.
- **Lasso regression** is especially useful for problems, where

> *the number of available covariates is extremely large, but only a handful of them are relevant for the prediction of the outcome.*

Lasso Regression

- Lasso regression is simply regression with L_1 penalty.
- That is, it solves the problem:

$$\hat{\beta}^* = arg \min_{\hat{\beta}} \sum_i \left(y^{(i)} - \hat{\beta} x^{(i)} \right)^2 + \lambda \|\hat{\beta}\|_1$$

- It turns out that the L_1 norm $\|\vec{x}\|_1 = \sum_j |x_j|$ **promotes sparsity**.
- The solution, $\hat{\beta}^*$, usually has only a small number of non-zero coefficients.
- The number of non-zero coefficients depends on the choice of the tuning parameter, λ. The higher the λ the fewer non-zero coefficients.

`glmnet`

- Lasso regression is implemented in an R package `glmnet`.
- An introductory tutorial to the package can be found here.

```
# install.packages("glmnet")
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-13
```

- We go back to `mtcars` datasets and use Lasso regression to predict the `mpg` using all variables.
- Lasso will pick a subset of predictors (the ones with non-zero coeffcents) that best predict the `mpg`.

```r
head(mtcars)
```

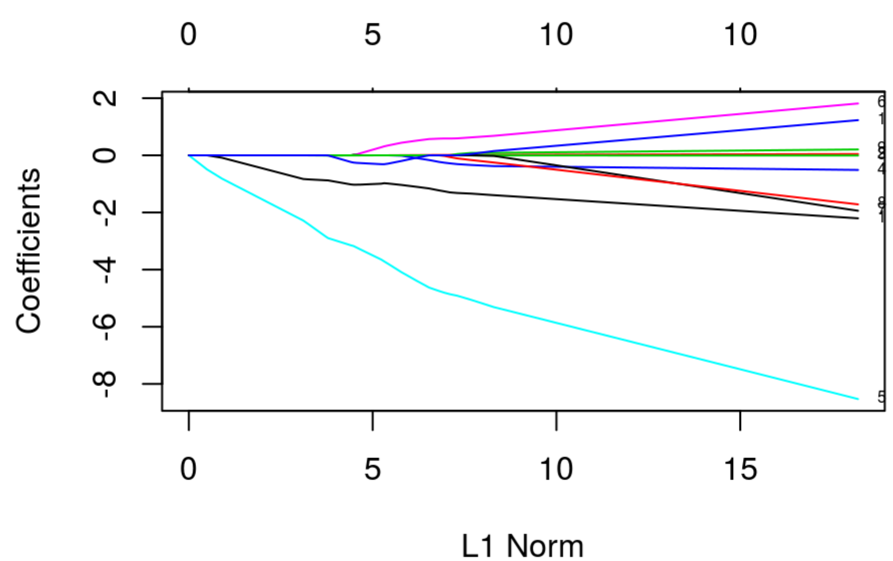
```
##    mpg cyl disp  hp drat    wt  qsec vs        am gear carb
## 1 21.0   6  160 110 3.90 2.620 16.46  0    manual    4    4
## 2 21.0   6  160 110 3.90 2.875 17.02  0    manual    4    4
## 3 22.8   4  108  93 3.85 2.320 18.61  1    manual    4    1
## 4 21.4   6  258 110 3.08 3.215 19.44  1 automatic    3    1
## 5 18.7   8  360 175 3.15 3.440 17.02  0 automatic    3    2
## 6 18.1   6  225 105 2.76 3.460 20.22  1 automatic    3    1
```

```r
y <- mtcars[, 1]  # mileage per gallon
x <- mtcars[, -1] # all other variables treated as predictors
x <- data.matrix(x, "matrix") # converts to NUMERIC matrix
# Choose a training set
set.seed(123)
trainIdx <- sample(1:nrow(mtcars), round(0.7 * nrow(mtcars)))
fit <- glmnet(x[trainIdx, ], y[trainIdx])
names(fit)
```

```
##  [1] "a0"        "beta"      "df"        "dim"       "lambda"
##  [6] "dev.ratio" "nulldev"   "npasses"   "jerr"      "offset"
## [11] "call"      "nobs"
```

- `glmnet()` compute the Lasso regression for a sequence of different tuning parameters, λ.
- Each row of `print(fit)` corresponds to a particular λ in the sequence.
- column `Df` denotes the number of non-zero coefficients (degrees of freedom),
- `%Dev` is the percentage variance explained,
- `Lambda` is the value of the currently chosen tuning parameter.

```r
# label = TRUE makes the plot annotate the curves with the corresponding coeffien
plot(fit, label = TRUE)
```



- the y-axis corresponds the value of the coefficients.
- the x-axis is denoted "L_1 norm" but is scaled to indicate the number of non-zero coefficients (the effective degrees of freedom).

- Each curve corresponds to a single variable, and shows the value of the coefficient as the tuning parameter varies.
- $||\hat{\beta}||_{L_1}$ increases and λ decreases from left to right.
- When λ is small (right) there are more non-zero coefficients.

The computed Lasso coefficient for a particular choice of λ can be printed using

```r
# Lambda = 1
coef(fit, s = 1)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept) 34.877093111
## cyl         -0.867649618
## disp         .
## hp          -0.005778702
## drat         .
## wt          -2.757808266
## qsec         .
## vs           .
## am           .
## gear         .
## carb         .
```

- Like for `lm()`, we can use a function `predict()` to predict the `mpg` for the training or the test data.
- However, we need specify the value of λ using the argument `s`.

```
# Predict for the test set:
predict(fit, newx = x[-trainIdx, ], s = c(0.5, 1.5, 2))
```

```
##              1        2        3
##  [1,] 25.36098 23.87240 23.22262
##  [2,] 19.82245 19.42427 19.41920
##  [3,] 16.19324 17.27111 17.74858
##  [4,] 22.62471 21.86937 21.50396
##  [5,] 15.20595 16.16123 16.71324
##  [6,] 11.25687 13.28117 14.26985
##  [7,] 10.81730 13.01570 14.07314
##  [8,] 25.88928 24.20103 23.47110
##  [9,] 27.01880 25.08206 24.22690
## [10,] 24.89106 23.51713 22.92237
```

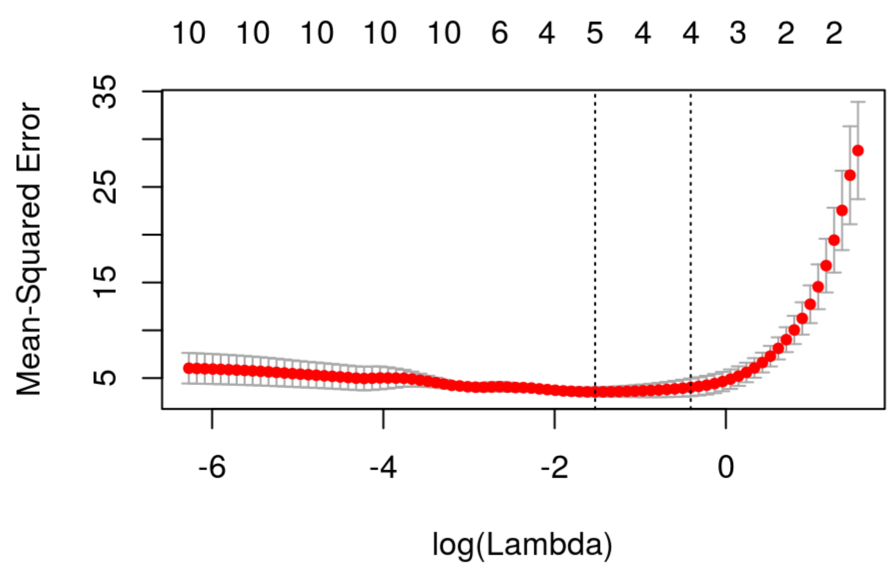
Each of the columns corresponds to a choice of λ.

Choosing λ

- To choose λ can use cross-validation.
- Use `cv.glmnet()` function to perform a k-fold cross validation.

> *In k-fold cross-validation, the original sample is randomly partitioned into k equal sized subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining $k - 1$ subsamples are used as training data.* [1]

```r
set.seed(1)
# `nfolds` argument sets the number of folds (k).
cvfit <- cv.glmnet(x[trainIdx, ], y[trainIdx], nfolds = 5)
plot(cvfit)
```



- The red dots are the average MSE over the k-folds.
- The two chosen λ values are the one with MSE_{min} and one with $MSE_{min} + sd_{min}$

λ with minimum MSE:

```
cvfit$lambda.min
```

```
## [1] 0.2171905
```

The biggest λ such that the MSE is within one standard error of the minimum M

```
cvfit$lambda.1se
```

```
## [1] 0.6632685
```

Extra Exercise

In this exercise you will perform Lasso regression yourself. We will use the Bos dataset from the MASS package. The dataset contains information on the Bostc suburbs housing market collected by David Harrison in 1978.

We will try to predict the median value of of homes in the region based on its attributes recorded in other variables.

First install the package:

```
# install.packages("MASS")
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:plotly':
##
##     select
```

```r
head(Boston, 3)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83
##   lstat medv
## 1  4.98 24.0
## 2  9.14 21.6
## 3  4.03 34.7
```

```r
str(Boston)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 .
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

Split the data to training and testing subsets.

```
set.seed(123)
trainIdx <- sample(1:nrow(Boston), round(0.7 * nrow(Boston)))
boston.test <- Boston[-trainIdx,"medv"]
```

Perform a Lasso regression with `glmnet`. Steps:

1. Extract the input and output data from the `Boston` `data.frame` and convert them if necessary to a correct format.
2. Use cross-validation to select the value for λ.
3. Inspect comuted coefficients for `lambda.min`.
4. Compute the predictions for the test dataset the two choices of the tuning parameter, `lambda.min` and `lambda.1se`. Evaluate the MSE for each.

1. https://en.wikipedia.org/wiki/Cross-validation_(statistics)#k-fold_cross-validation↩