



ELEMENTS OF DATA SCIENCE AND STATISTICAL LEARNING

SPRING 2017

Week 3

CLASS INFO

- Online lecture videos by Trevor Hastie and Rob Tibshirani:

<http://www.dataschool.io/15-hours-of-expert-machine-learning-videos/>

- Leo Breiman's famous paper:

<https://projecteuclid.org/euclid.ss/1009213726>

OUTLINE

- Recap of a few topics from last week
- Maximum Likelihood Estimation (MLE) :“predicting” the value of a single variable
- Case of two variables: a simple simulation
- Linear Regression and MLE
- Accuracy and significance of the model parameter estimates, accuracy of the model
- Categorical variables, multiple variables

VARIABLE SCOPING

- The body of if/else, while or for statement does NOT have its own scope (variable first defined inside if/else or a loop is available afterwards)
- Variables defined inside functions (including formal parameters) are local to those functions and will mask variables with the same names in the outer scope
- Outer scope (“global”) variables are visible inside function body (unless masked by another variable) but unmodifiable (sort of)
 - Global variables are “copy-on-change”; it is possible to modify a global variable (and even the argument passed to a function!) from inside a function, but you really shouldn’t!

```
> y=1:3
> f=function() {
  print(y)
  y[2]=5 # now y is local!
  print(y)
}
> f()
[1] 1 2 3
[1] 1 5 3
> y
[1] 1 2 3
```

```
> y=1:3
> f=function() {
  print(y)
  y[2]<-5 # assign in outer scope
  print(y)
}
> f()
[1] 1 2 3
[1] 1 5 3
> y
[1] 1 5 3
```

LAZY EVALUATION

- Evaluations in R are performed lazily (only when needed)
 - Logical expressions are short-circuited (something you most likely have seen before)
 - Function arguments and even *default values* are not evaluated until they are needed!

```
> x=5
# need to evaluate both expressions:
> ! is.na(x) && x > 0
[1] TRUE
> x=NA
# evaluates only part to the left of &&:
> ! is.na(x) && x > 0
[1] FALSE
# proof: undefined function - no problem!
> ! is.na(x) && some.undefined.fun(x)
[1] FALSE
# Can you explain what happened here:
> F & some.undefined.fun(x)
Error: could not find function
"some.undefined.fun"
```

```
f=function(x,y=sqrt(z)) {
  if ( x == 1 ) print("I need only x")
  else {
    if ( x == 2 ) { y=y+1; print("I need x and y") }
  }
}
> f(1) # y never used, default not evaluated!
[1] "I need only x"
> f(1,sqrt(W)) # y not used, passed arg not evaluated!
[1] "I need only x"
> f(2) # needs y for y=y+1, cannot evaluate default!
Error in f(2) : object 'z' not found
> z=4 # now z is defined...
> f(2) # and the function can evaluate the default for y!
[1] "I need x and y"
> f(2,y=sqrt(W)) # arg is evaluated since y is needed
Error in f(2, y = sqrt(W)) : object 'W' not found
```

TRADITIONAL LOOPS VS VECTORIZED CODE

- A word on vectorized arithmetic (cannot be overstated):

```
> x<-numeric(10000000)
> system.time(x[] <- 1)
  user  system elapsed 
0.03    0.00    0.03 
> system.time(for(i in 1:length(x)) x[i] <- 1)
  user  system elapsed 
10.44    0.02   10.54
```

- R provides mapping functions (the 'apply' family); contrary to popular belief they are not necessarily faster, but at least they are convenient and fit well into the whole paradigm of vectorized calculations:

```
> y=numeric(length(x))
> system.time(for(i in 1:length(x)) y[i]=x[i]^2)
  user  system elapsed 
16.22    0.00   16.27 
> system.time(y <- apply(x,FUN=function(x) x^2 ))
  user  system elapsed 
18.82    0.06   18.91 
> system.time(y <- x^2)
  user  system elapsed 
0.02    0.00    0.02
```

Why can't we use '='?

```
> m=matrix(1:6,ncol=2)
> m
      [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
# apply a function to each row (MARGIN=1);
# to apply to each column we'd use MARGIN=2:
> apply(m,MARGIN=1,sum)
[1] 5 7 9
```

FACTORS

- R has yet another basic data type that we ignored so far: factors
 - In statistics, a “factor” is another name for a categorical variable, the one that takes on a finite set of fixed values
 - Thus the ‘factor’ data type is most similar to what in many other languages is called ‘enum’ (enumeration type)
 - Very useful and many functions know how to use factors correctly. NOTE: implementation may result in very unexpected consequences if you are not careful when using factors
 - Dataframes convert character vectors (columns) into factors *automatically* – both when explicit constructor `data.frame()` is used and when a data frame is loaded with `read.table` – sometimes this is NOT what we want (e.g. patient’s gender, M/F is a bona fide categorical variable, while patient’s name is probably nothing but just an attribute with no particular statistical significance). If `p.names` is a character vector, we can use `I()` in the constructor (e.g. `data.frame(patient.name=I(p.name),...)` to prevent the conversion, and in `read.table` we can use `as.is=T` (applies to ALL character columns!)

```
> class(iris)
[1] "data.frame"
> sapply(iris,mode)
Sepal.Length Sepal.Width Petal.Length Petal.Width Species
"numeric"    "numeric"    "numeric"    "numeric"    "numeric"
> sapply(iris,class)
Sepal.Length Sepal.Width Petal.Length Petal.Width Species
"numeric"    "numeric"    "numeric"    "numeric"    "factor"
```

WORKING WITH FACTORS

- A few examples of working with factors:

```
> s<-c("M", "F", "M")
> f<-as.factor(s); f
[1] M F M
Levels: F M
> as.numeric(s)
[1] NA NA NA
Warning message:
NAs introduced by coercion
> as.numeric(f)
[1] 2 1 2
> levels(f)
[1] "F" "M"
> as.character(f)
[1] "M" "F" "M"
> f[3]<-"Z"; f
Warning message:
In `[<-.factor`(`*tmp*`, 3, value = "Z") :
invalid factor level, NAs generated
[1] M F <NA>
```

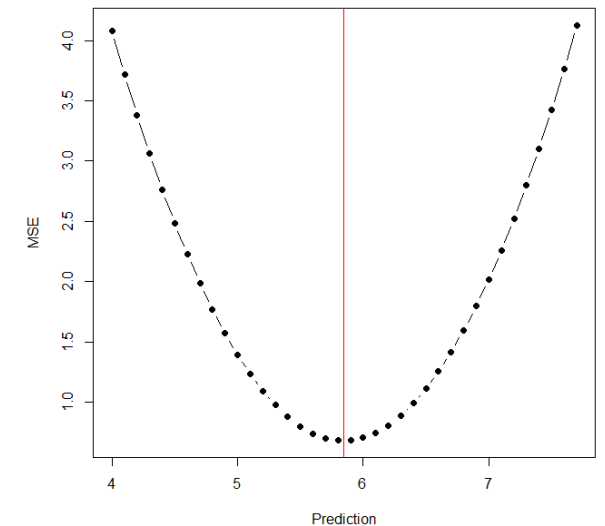

PREDICTION PROBLEM

- What does it mean to “predict” anything (in statistical sense)?
 - There is a quantity of interest, an *outcome, response, or dependent variable* (all interchangeable), often denoted as Y (it is a random variable!)
 - We have some “historical” (previously measured) values (a realization) y_1, y_2, \dots, y_k
 - We want to know what the next measurement is going to be:
 $Y = [\text{prediction generator}]$
 - Our “prediction generator” gives some value(s) \bar{y} . Ideally, we would like to have $y - \bar{y} = 0$, for all measurements, past and future. Not going to happen, most of the time. We can characterize predictions by their accuracy, i.e. how far they are from true values. For instance, $\frac{1}{k} \sum_i (y_i - \bar{y}_i)^2$ - this quantity is known as mean squared error, or MSE.
 - Our goal is to produce “the best” prediction generator. When we use previously measured data – we are “learning from example”
 - In most (all?) practical settings, we have other variables measured, X_1, \dots, X_n , and we want to use those to help us with prediction

PREDICTION AND VARIANCE: SINGLE VARIABLE

- Try to predict the following:
 - Will the sun rise tomorrow?
 - What is the result of the next coin toss?
- Probability distribution is all we need to make a prediction. What is the best prediction we can make if the distribution of Y is all we know (i.e. we do not measure anything else)?
 - Try predicting the height of the next person to come into this room
 - Try predicting the sepal length of the next iris plant you are going to see

```
> mse=numeric()  
> for ( x in seq(4,7.7,by=0.1 ) ) {  
  mse=c(mse,sum( (x-iris$Sepal.Length)^2)/150)  
}  
> plot(seq(4,7.7,by=0.1 ),mse,xlab="Prediction",ylab="MSE",pch=19,type="b")  
> abline(v=mean(iris$Sepal.Length),col="red")
```

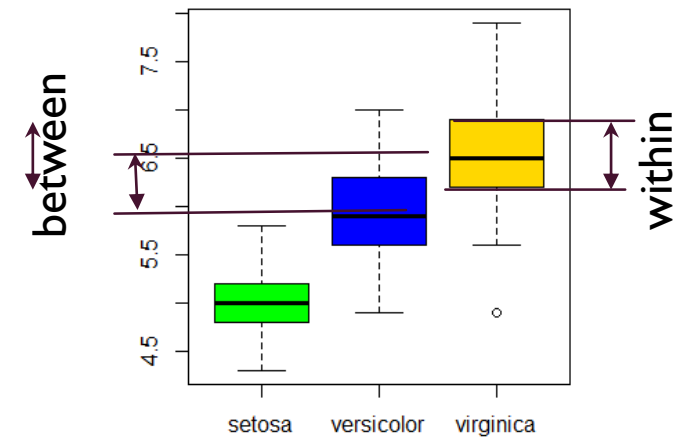


- Note: if we defined the accuracy as $\frac{1}{k} \sum_i |y_i - \bar{y}_i|$ the best prediction would be median!

PREDICTION AND VARIANCE: EXPLANATORY VARIABLES

- Now let's recollect that the sepal length seem to depend on the species
- Now, in the presence of the “explanatory variable” we can predict much better!
 - What our predictions are going to be for each species?
- Let us look at sums of squares in different scenarios:

```
> v.total = var(iris$Sepal.Length)
> levels(iris$Species)
[1] "setosa"      "versicolor" "virginica"
> v.setosa = var(iris$Sepal.Length[iris$Species=="setosa"])
> v.versi = var(iris$Sepal.Length[iris$Species=="versicolor"])
> v.virg = var(iris$Sepal.Length[iris$Species=="virginica"])
> m1=mean(iris$Sepal.Length[iris$Species=="setosa"])-mean(iris$Sepal.Length)
> m2=mean(iris$Sepal.Length[iris$Species=="versicolor"])-mean(iris$Sepal.Length)
> m3=mean(iris$Sepal.Length[iris$Species=="virginica"])-mean(iris$Sepal.Length)
> v.total*149
[1] 102.1683
> (v.setosa+v.versi+v.virg)*49 + (m1^2+m2^2+m3^2)*50
[1] 102.1683
```



We used the fact that
 $\text{variance} = \frac{\sum (\text{squared diff from mean})}{(\text{Nobservations} - 1)}$

Note: the figure is an illustration only as boxplots show medians/IQRs, not means/variances

PREDICTION AND VARIANCE: CONTINUED

- What we observed (can be proved, of course) is that the squared sum of “errors” in the best possible prediction without explanatory variables (which is just the sample mean) is equal to the sum of remaining errors in predictions *with* explanatory variable PLUS the squared differences between new predictions and the sample mean.
- Variation within each species (“within” term) is still unexplained – even if we know the species, we cannot predict better than just the mean *in that subset*
- Variation between different predictions (means of the subsets) and the sample mean (“between” term) is what we *explained* : indeed, this is what we *predict* based on the species!
- Total variation in the dataset = Unexplained (remaining) variation + Explained (predicted) variation
- Or in other words prediction = reducing remaining unexplained variance!

STATISTICAL LEARNING

- Previously, we have written the “idea” of prediction down as $Y = [\text{prediction generator}]$
- More traditional and formal way is to write $Y = f(X) + \varepsilon$
 - Here X are all the explanatory variables we are considering, X_1, \dots, X_n
 - $f(X)$ is the “prediction generator” – it does not necessarily have a closed functional form (e.g. $f(x) = Ax + B$), but of course it should be computable in some way, e.g. by computer program. For each measurement (realization) x_1, \dots, x_n of the random processes X , we can substitute those values into the prediction generator and we get back the corresponding predicted value.
 - The predicted value is not going to be always precisely correct – there will be *unexplained* variance remaining, which is described by the noise term ε .
 - Note that we call “noise” everything that our model does not explain: it can be “true” random noise or it can be some effect that *could* be explained if we could (or cared to) include additional variables into our model! Return to our Sepal Length example: if you examine the pairwise scatterplots carefully, you could observe that *within* each species sepal length still correlates pretty well with sepal width. So we could improve our model further and explain a bigger chunk of the total variance in the observations of the sepal length! But for as long as we don’t include sepal width, all the remaining, “within” variance in our boxplots is just that: “noise”.

PREDICTION VS INFERENCE

- In prediction problem we are interested in predicting outcome in new cases
 - Stock price
 - Patient survival
- In inference problem we are more interested in describing the data at hand
 - Which parameters are important?
 - Which parameters have positive/negative effect?
- In general, there is no clear distinction between models that can be used for prediction or inference, but the properties of different models may make them more or less desirable in different settings
 - For instance, extremely complex model that does not allow for clear interpretation of the dependencies or relative importance of the variables might be something we are more willing to adopt for prediction, but not for inference

OVERFITTING

- The model can perform great on the training dataset, but does it generalize well?
- We can have so many parameters in the model/so few data points that our model starts fitting the noise!
- If prediction is what we are after, we **MUST** evaluate our model on a test dataset – a dataset that has **NOT** been used for training. Accuracy on the *test* dataset is of the greatest importance.
- Data that we are trying to fit a model to are only a *sample* (randomly drawn!). Just like sample mean is a random variable, so is any quantity that is computed from a given sample. Thus, it also includes a model (or rather its *parameters*)
 - Models that are too flexible have high **variance**: change a lot from one training dataset to the next, i.e. fit noise
- **Bias** is the error due to oversimplification of the model, i.e. the (expected value of the) difference between what the prediction could be if we knew the “true” dependence and the prediction that our model makes.
 - For instance, if the true dependence is $Y = a X^2 + b X$, but we are only considering a model $Y = b X$, we are having a bias.
 - Bias-Variance tradeoff: there is a sweet spot where overall accuracy is optimal
 - Note that the “true model” may or may not have any particular closed functional form. We refer to it here, somewhat informally, as some “true underlying dependence”.

OUTLINE

- Maximum Likelihood Estimation (MLE) :“predicting” the value of a single variable
- Case of two variables: a simple simulation
- Linear Regression and MLE
- Accuracy and significance of the model parameter estimates, accuracy of the model
- Categorical variables, multiple variables

MAXIMUM LIKELIHOOD: MOTIVATION

- Maximum likelihood is a convenient, powerful and unifying principle one can use to look at many statistical problems.
- Consider a general situation, when we have some data and a (parametrized) model $M(\theta)$, and we want to “fit” the latter to the data
 - “Fitting” the model means finding the “optimal” parameters (in some sense): those that “best describe the data”
 - The data is a random variable D (we sample randomly!), hence we can say that any particular experiment is a realization $D=d$.
 - The model parameters are fitted on the observed data: we can say that upon observing the data we choose *the most likely* parameters of the model. This suggests that we can speak of a *probability distribution/density* in the parameter space.
- Example: consider a mean.
 - We get a *sample* and its sample mean is 3.1415. We know for sure that the true population mean is not *exactly* 3.1415. Can it be equal to 3? 4? 27.5? What is the *most likely* value of the population mean *given the data* (in other words, what is the *estimate* we should settle upon) – the answer is of course “the sample mean”!

MAXIMUM LIKELIHOOD

- Consider a *joint* distribution of the data samples and model parameters, $P(\Theta, D)$
 - Of course we do hope that any particular measurement of the data tells us something about the parameters we should choose, so the variables are not independent, $P(\Theta, D) \neq P(\Theta)P(D)$, that's fine
- We can always write $P(\Theta, D) = P(\Theta|D)P(D) = P(D|\Theta)P(\Theta)$, in other words

$$P(\Theta|D) = \frac{P(D|\Theta)P(\Theta)}{P(D)} \propto P(D|\Theta) \quad (\text{Bayes' Theorem!})$$

 Likelihood function

- Maximum Likelihood Estimation (MLE) amounts to choosing $\Theta = \theta$ which *maximizes the likelihood function*
- Try to interpret
 - Why do we discard $P(D)$ and $P(\Theta)$? What does it mean?
 - What's the interpretation of $P(\Theta|D) \propto P(D|\Theta)$?

MAXIMUM LIKELIHOOD: INTERPRETATION

- Data are *given*, whatever value $P(D=d)$ takes, it is just a constant; it also does not depend on Θ [one could also argue that if someone tells us that the experimental data we got are not representative at all, we probably should throw the experiment out and try again; but in the absence of any such knowledge we have to *assume* that the dataset we got is as good as any; indeed, if we were unlucky enough to get a very unusual sample and, for instance, use its mean as the population mean estimate, we'd be way off, there is no way around it!]
- Discarding $P(\Theta)$ is justified by saying that we have no strong feeling for one value of parameter(s) over another before running an experiment (“uniform prior”)
 - But if we do have previous knowledge that favors some parameter values over the others, we can keep the $P(\Theta)$ term. By doing that we enter the realm of Bayesian statistics, $P(\Theta)$ is called a “prior”, and $P(\Theta|D)$ is called the “posterior”.
- Interpretation of the likelihood function: in order to find out the optimal value θ of model parameter(s), *given the data* (!), we have to calculate how likely it would be to *observe* such data *if* the parameters were indeed equal to θ .

$$P(\Theta|D) \propto P(D|\Theta)$$

MAXIMUM LIKELIHOOD: MEAN AS AN EXAMPLE

- Suppose we have a normally distributed variable Y ; what is the MLE estimation of the “location”?
 - Remember the earlier example where we observed that the mean looks like the best “prediction” we can come up with in the “world” where Y is the only thing ever measured
 - How much time does it take you to get home after the lecture? Can you predict how much time it is going to take you tonight? “22 minutes give or take” = the best prediction (the mean) + error (how many traffic lights are going to be red? Is there an accident along the way?)
 - In terms of a “model” we thus say $Y = \beta_0 + \varepsilon$ (where β_0 is some constant) – we don’t have any “explanatory variables” X available.
- The error is *normally distributed*: $P(\varepsilon = e) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{e^2}{2\sigma^2}\right)$ (it has to be since that’s the only random term we have and Y is normal); but for any measurement y_i we take, the error of our model is $e_i = y_i - \beta_0$, hence the probability to observe data, *given the parameters* is

Likelihood

$$L = P(D|\Theta) = \prod_i P(e_i) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^N \exp\left[-\frac{1}{2\sigma^2} \sum_i (y_i - \beta_0)^2\right]$$


Independent errors!

Errors normally distributed, with fixed variance (normal i.i.d)!

MAXIMUM LIKELIHOOD: MEAN AS AN EXAMPLE, CONTINUED

- According to Bayes theorem probability density of the parameter β_0 (likelihood) is proportional to $P(D | \beta_0)$ shown above
- According to the Maximum Likelihood Principle, we should run with the most likely value for β_0
 - Bringing these two statements together, we should find the maximum of $P(D | \beta_0)$ as a function of β_0 (the data are measured and *fixed*).
- It is convenient to take the Log of the likelihood function (does not change anything as far as maximum is concerned: log is a monotonous function, but math becomes easier):

$$\log L = \log P(D | \Theta) = N \log \left(\frac{1}{\sqrt{2\pi} \sigma} \right) - \frac{1}{2\sigma^2} \sum_i (y_i - \beta_0)^2 \quad (\text{thus max log L is min of } \sum (y_i - \beta_0)^2)$$

- At an extremum, the derivative must be 0: $\frac{\partial \log L}{\partial \beta_0} = 0$  leads to the estimate $\hat{\beta}_0 = \frac{1}{N} \sum y_i = \bar{y}$
- According to ML principle, sample average is the best predictor of the population mean

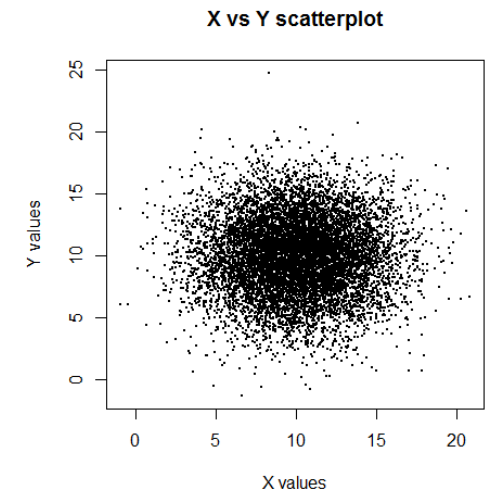
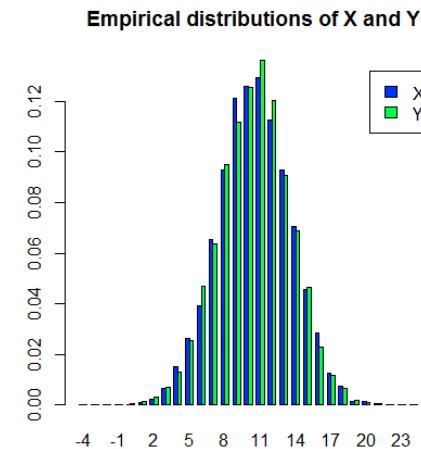
OUTLINE

- Homework I Solutions
- Maximum Likelihood Estimation (MLE) :“predicting” the value of a single variable
- **Case of two variables: a simple simulation**
- Linear Regression and MLE
- Accuracy and significance of the model parameter estimates, accuracy of the model
- Categorical variables, multiple variables

CASE OF (TWO) INDEPENDENT VARIABLES

- Suppose we have two *independent* normal variables X,Y

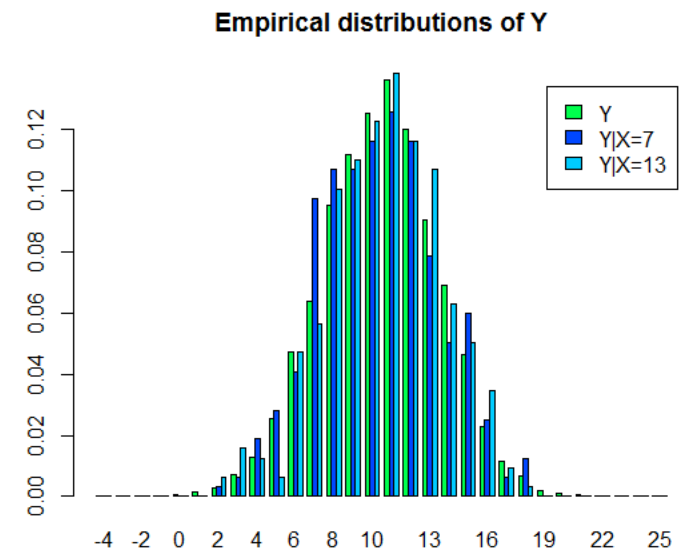
```
# simulate sampling of 10000 values for X and for Y:
> x <- rnorm(10000,mean=10,sd=3)
> y <- rnorm(10000,mean=10,sd=3)
> br<- -5:25 # set manually bins for histograms
# save histograms for X and Y , don't plot yet:
> hx <- hist(x,breaks=br,plot=F)
> hy <- hist(y,breaks=br,plot=F)
# prepare 2 panels in one plot:
> old.par <-par(mfrow=c(1,2))
# plot histograms side by side (they better have same
# bins, but we ensured that):
> barplot(rbind(hx$density,hy$density),beside=T,
          col=c(rgb(0,0.2,1),rgb(0,1,0.3)),legend=c('X','Y'),
          main='Empirical distributions of X and Y',
          names=br[-1])
> plot(x,y,xlab='X values',ylab='Y values',
       main='X vs Y scatterplot',pch=19,cex=0.3)
# restore graphical attributes to previous values:
> par(old.par)
```



STATISTICAL INDEPENDENCY

- We can look at conditional probability densities of Y at different X and of course they are the same (and same as $P(Y)$): $P(Y|X)=P(Y)$

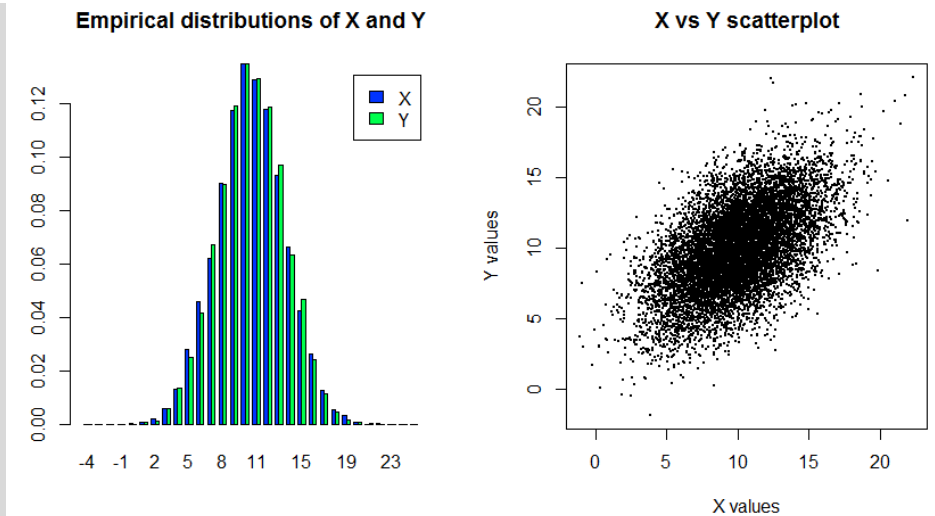
```
# select instances where X is close to 7 or to 13:
> range.7 <- x > 6.8 & x < 7.2
> range.13 <- x > 12.8 & x < 13.2
# take only Y where X~7 and calculate their empirical distribution:
> hy.7 <- hist(y[range.7],breaks=br,plot=F)
# select only Y where X~13 and calculate the empirical distribution:
> hy.13 <- hist(y[range.13],breaks=br,plot=F)
# plot overall distribution of Y and conditional distributions,
# P(Y|X=7) and P(Y|X=13) side by side
> barplot(rbind(hy$density,hy.7$density,hy.13$density) ,
          beside=T,col=c(rgb(0,1,0.3),rgb(0,0.28,1),rgb(0,0.8,1)) ,
          legend=c('Y','Y|X=7','Y|X=13') ,
          main='Empirical distributions of Y',names=br[-1])
> t.test(y[range.7],y[range.13])
Welch Two Sample t-test
...
t = -0.98274, df = 633.33, p-value = 0.3261
```



DEPENDENT VARIABLES

- Let us now simulate two variables with a built-in dependency:

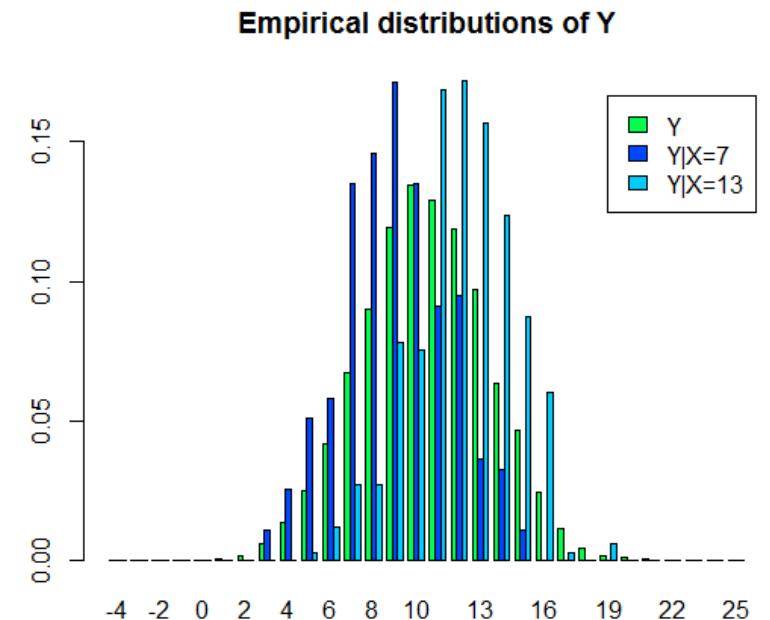
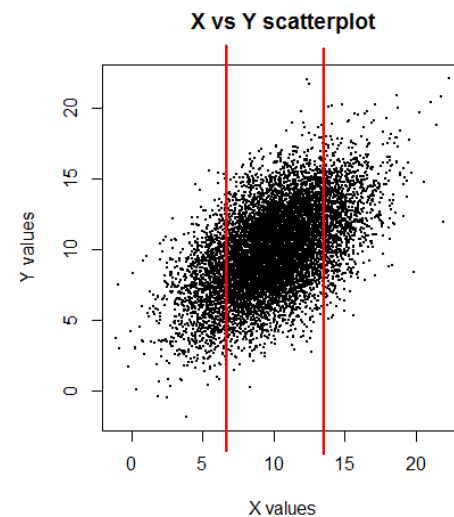
```
> x <- rnorm(10000,mean=10,sd=sqrt(5))
> y <- x # initialize y with x
> x <- x + rnorm(10000,sd=2)
> y <- y + rnorm(10000,sd=2)
> br<- -5:25 # set manually bins for histograms
> hx <- hist(x,breaks=br,plot=F) # save histogram of X, don't plot
> hy <- hist(y,breaks=br,plot=F) # save histogram of Y, don't plot
> old.par <- par(mfrow=c(1,2)) # prepare to draw 2 plots in one row
# now plot histograms side by side (we ensured they use same bins):
> barplot(rbind(hx$density,hy$density),beside=T,
          col=c(rgb(0,0.2,1),rgb(0,1,0.3)),legend=c('X','Y'),
          main='Empirical distributions of X and Y',names=br[-1])
> plot(x,y,xlab='X values',ylab='Y values',
       main='X vs Y scatterplot',pch=19,cex=0.3)
> par(old.par) # restore graphical attributes to previous values
```



CONDITIONAL DISTRIBUTIONS IN CASE OF DEPENDENCY

- Finally, let us look at the same conditional probability distributions we defined earlier, but this time for our redefined, dependent variables X, Y

```
> range.7 <- x > 6.8 & x < 7.2
> range.13 <- x > 12.8 & x < 13.2
> hy.7 <- hist(y[range.7], breaks=br, plot=F)
> hy.13 <- hist(y[range.13], breaks=br, plot=F)
> barplot(rbind(hy$density, hy.7$density, hy.13$density),
  beside=T, col=c(rgb(0,1,0.3), rgb(0,0.28,1), rgb(0,0.8,1)),
  legend=c('Y', 'Y|X=7', 'Y|X=13'),
  main='Empirical distributions of Y', names=br[-1])
```



HOW TO ESTIMATE/PREDICT Y?

- In general, $P(Y|X)$ is everything we need to (or can!) know
- Everything we do in statistical learning amounts to
 - Finding a (sub)set or transformation of variables X such that $P(Y|X)$ conditioned on them is as narrow as possible
 - Approximating $P(Y|X)$
- Note that in our trivial example we had so many data points available that we could directly estimate $P(Y|X)$ by using reasonably narrow bins in the domain of X !
 - Note also that this might work fine for prediction (albeit inefficient), but does not summarize the data in any way
 - Remember that the particular toy dataset was simulated with linear dependency between Y and X *explicitly built in*.
 - At each X we can only predict mean Y (based on $P(Y|X)$ of course). If that mean traces linearly with x , wouldn't it be nice to discover that and use for prediction (note that anything more complex than that is *overfitting* – the benefit of using a simulated dataset is knowing *exactly* what's in the data and what is not).

OUTLINE

- Homework I Solutions
- Maximum Likelihood Estimation (MLE) :“predicting” the value of a single variable
- Case of two variables: a simple simulation
- Linear Regression and MLE
- Accuracy and significance of the model parameter estimates, accuracy of the model
- Categorical variables, multiple variables

LINEAR REGRESSION

- Consider the situation when we do have additional measurements that (hopefully) can guide our predictions
 - Time of the day when you are going to drive home (6pm? 8pm? 11:30pm?) – if it's known, you'd probably make a better prediction of the duration of your trip
- One of the simplest possible models that includes explanatory variable(s) is a *linear regression model*:

$$Y = \beta_0 + \beta_1 X + \varepsilon$$

- Or, as it is often denoted in statistical texts (does it remind you something we have seen in R?):

$$Y \sim X + \varepsilon \text{ or even } Y \sim X$$

- The meaning is straightforward:
 - For each measured x_i , the *predicted* value of Y is $\hat{y}_i = \beta_0 + \beta_1 x_i$. The difference between the true value and prediction is the catch-all “noise” (measurement error, model bias, other variables), with zero mean:

$$e_i \equiv y_i - \hat{y}_i = y_i - \beta_0 - \beta_1 x_i$$

Residual of the model

LEAST SQUARES

- Our goal is to choose the coefficients β_0, β_1 in such a way that our predictions are as close as possible to the observed values.
- The most common approach involves minimizing the residual sum of squares (RSS):

$$RSS = \sum_i e_i^2 = \sum_i (y_i - \beta_0 - \beta_1 x_i)^2$$

- This is the *least squares method*. Why least squares? MLE is the answer again:

$$L = P(D|\Theta) = \prod_i P(e_i) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^N \exp\left[-\frac{1}{2\sigma^2} \sum_i (y_i - \beta_0 - \beta_1 x_i)^2\right]$$

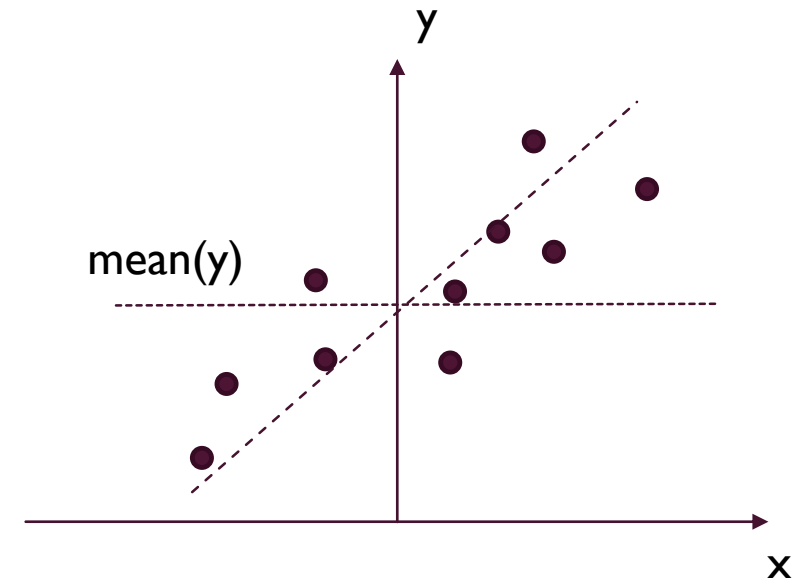
← This is RSS! Minimize to get max. L!

- The least squares criterion has a very specific meaning! It is the result of assuming normal i.i.d. noise!
- Estimates:

$$\hat{\beta}_1 = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}; \quad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

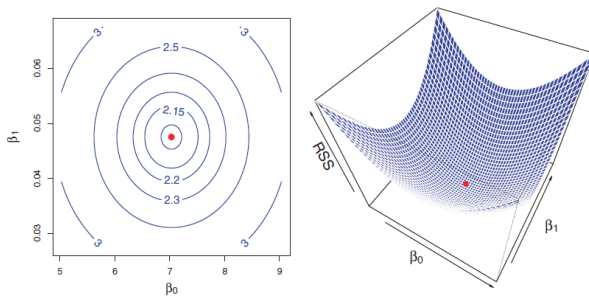
MEANING OF THE INTERCEPT

- We have: $\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$
- If the variable X is centered, ($\bar{x}=0$), then $\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} = \bar{y}$
 - The intercept is the “overall” mean of the outcome variable Y ; the linear term in the model describes by how much we expect the measurements of Y to deviate from that mean depending of X
- If X is not centered, then we just add a (constant) shift to the intercept; it still characterizes the (total) mean of Y .



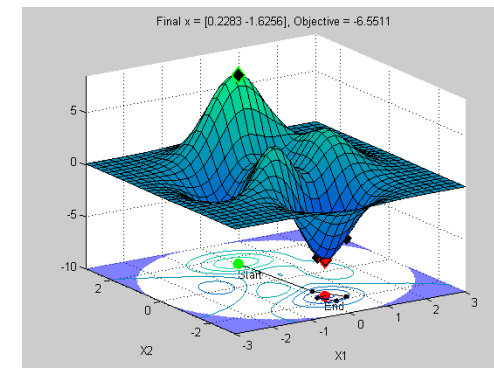
A NOTE ON PARAMETER OPTIMIZATION

- MLE provides a conceptual framework for finding optimal parameters in any model
- Consider the (log-)likelihood, $\log L = f(\theta)$: a standard (albeit not necessarily trivial) problem of finding a minimum of a function (in multi-dimensional space, strictly speaking).
- The example we have seen (linear model) is trivial: there is an analytic solution



$$\leftarrow -\log L \propto \left[\sum_i (y_i - \beta_0 - \beta_1 x_i)^2 \right]$$

- In general, the surface can be very complex for an arbitrary $f(\theta)$, have multiple local minima etc
 - Optimization problem (not the topic of this class!)
 - More advanced math helps sometimes
 - More advanced numerical algorithms help often
 - Ultimate case: Monte-Carlo simulation (with gradient descent), just sample the parameter space!



OUTLINE

- Homework I Solutions
- Maximum Likelihood Estimation (MLE) :“predicting” the value of a single variable
- Case of two variables: a simple simulation
- Linear Regression and MLE
- Accuracy and significance of the model parameter estimates, accuracy of the model
- Categorical variables, multiple variables

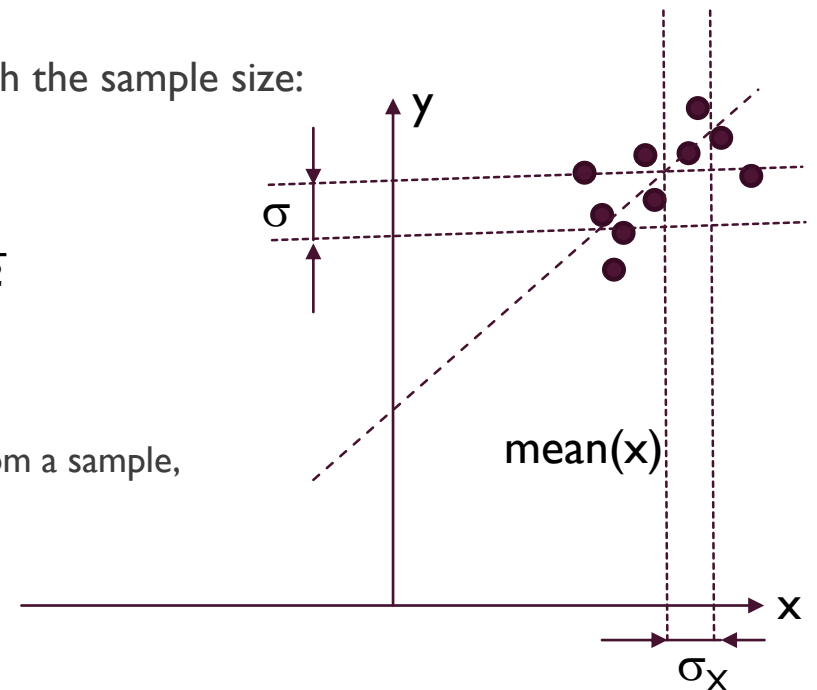
ACCURACY OF THE ESTIMATES

- The regression coefficient estimates $\hat{\beta}_0, \hat{\beta}_1$ are estimated from the observed data (particular randomly drawn realization!), hence they are also random variables!
 - Earlier we noted that sample mean (or in fact any sample statistic) is a random variable – the present case is exactly the same!
 - You may remember from the homework that $\text{Var}(\hat{\mu}) = \text{SE}(\hat{\mu})^2 = \sigma^2/n$, where σ^2 is the variance of the distribution we are sampling from
 - Similarly, the accuracy of the estimates of the linear model coefficients improves with the sample size:

$$\text{SE}(\hat{\beta}_0)^2 = \sigma^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \right]$$

$$\text{SE}(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

- But what is the standard deviation, σ , of the error?
 - Just like we estimated the mean *and* the standard deviation of the underlying distribution from a sample, we can do it here as well: $\hat{\sigma} = \text{RSE} = \sqrt{\text{RSS}/(n-2)}$ ← **Residual standard error**
 - NOTE that the estimate is *pooled* across all observations!

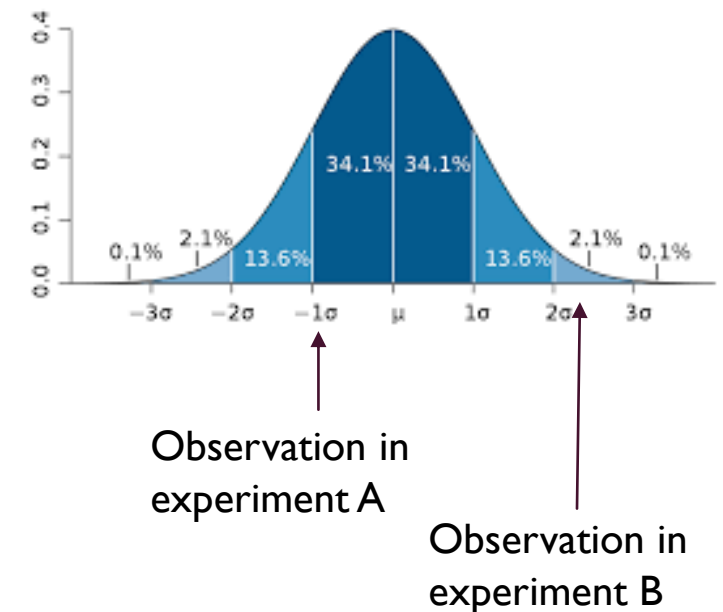


SIGNIFICANCE OF THE ESTIMATES

- We know the typical error associated with our predictions. But are those predictions *significant* at all
- To answer this question we should *test a hypothesis*
 - Namely, what if there is no association between X and Y (true slope $\beta_1=0$), and we observed non-zero slope solely due to random sampling error? This is our *null hypothesis*: $\beta_1=0$.
 - *If* the null were true, what chance do we have to (randomly) observe the slope at least as extreme as the one that resulted from our experiment?

$$\hat{\beta}_1 = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}$$

- How is the *random variable* β_1 distributed?
- If we knew the variance of β_1 precisely, we would just use the *standardized* value of the slope (in units of its standard deviation) and read the answer from the picture on the left (normal distribution is shown there)
- In practice, standard deviation is itself an estimate ($\hat{\sigma} = \text{RSE} = \sqrt{\text{RSS}/(n-2)}$). As the result, $\beta/\hat{\sigma}$ (a ratio of two random variables!) is distributed not (quite) normally. This distribution is known as t-distribution, and this is the correction t-test accounts for (it performs the procedure as depicted on the right, but with corrected distribution)

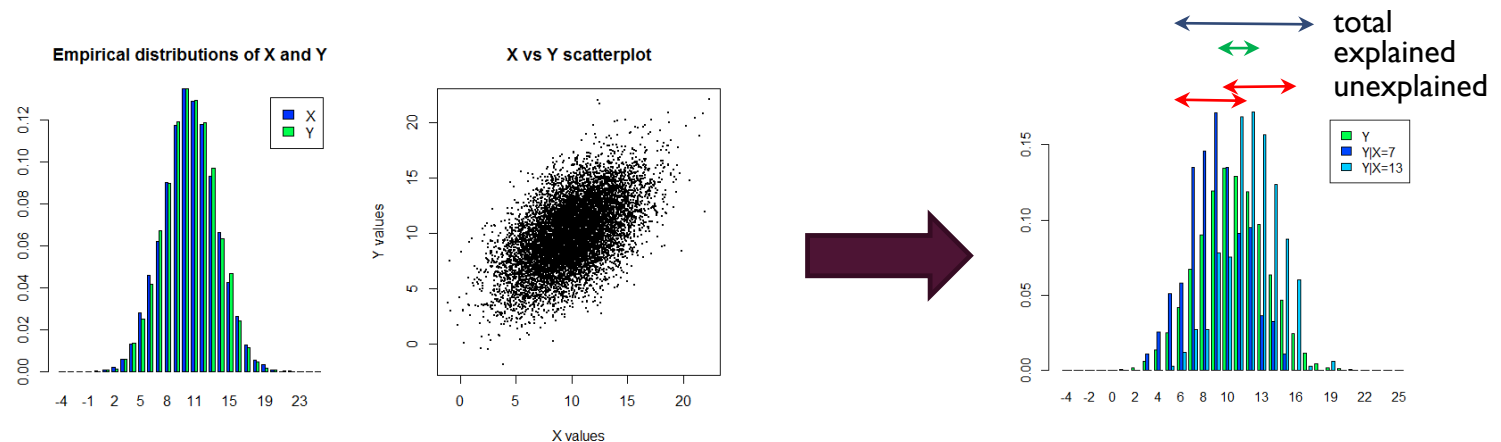


ACCURACY OF THE MODEL

- How well we can calculate the coefficients (and how significant they are) is one question
- How well the resulting model *explains the data* is a completely different question!
 - We can use RSS or RSE: measures absolute lack of fit (total or average per experimental point, respectively). But what value of RSE is “good”?
 - R^2 : the *proportion* of variance explained by the model. Total sum of squares, $TSS = \sum_{i=1}^n (y_i - \bar{y})^2$, then

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}$$

- RSS is the *remaining, unexplained* variance
- Note that in simple linear regression, R^2 is equal to the correlation coefficient, squared: $R^2 = r^2$.



LINEAR REGRESSION IN R

- Let's start with a simulated example, where we know the answer:

```
> x=seq(-3,3,length=100)
> y=1+2*x+rnorm(100,sd=1)
> plot(x,y,pch=19,cex=0.7)
> m=lm(y~x) # this is all it takes!
> summary(m)
```

Call:

```
lm(formula = y ~ x)
```

Residuals:

Min	1Q	Median	3Q	Max
-2.9551	-0.7838	-0.1293	0.8964	2.5680

Coefficients:

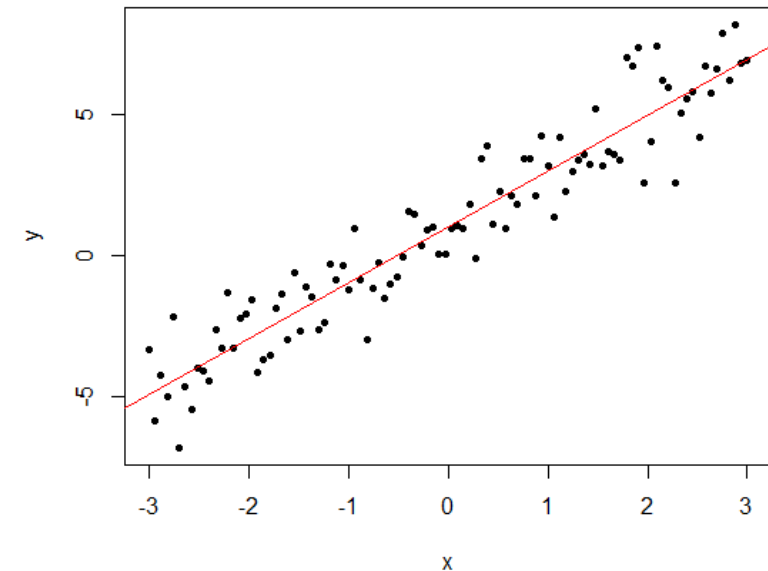
	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	1.00348	0.11523	8.709	7.62e-14 ***
x	1.98041	0.06586	30.068	< 2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.152 on 98 degrees of freedom

Multiple R-squared: 0.9022, Adjusted R-squared: 0.9012

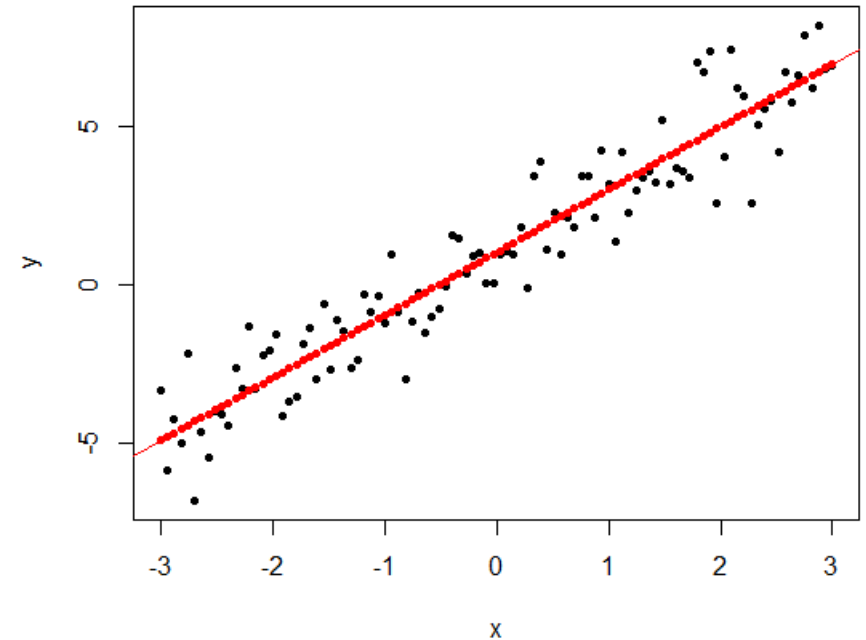
F-statistic: 904.1 on 1 and 98 DF, p-value: < 2.2e-16



EXPLORING THE METRICS

- Let us calculate a few metrics manually and compare them to the output of the linear model (previous slide)

```
# all it takes to get the predicted values
# for the SAME x used for training:
> yp=predict(m)
> points(x,yp,col="red",pch=19,cex=0.7)
> rss=sum((y-yp)^2)
> rss
[1] 130.1152
> rse=sqrt(rss/98)
> rse
[1] 1.152261
> tss=sum((y-mean(y))^2)
> tss
[1] 1330.486
> (tss-rss)/tss
[1] 0.9022048
# let's compute manually standard error of the slope:
> rse/sqrt(sum((x-mean(x))^2))
[1] 0.06586386
```



PREDICTION ACCURACY

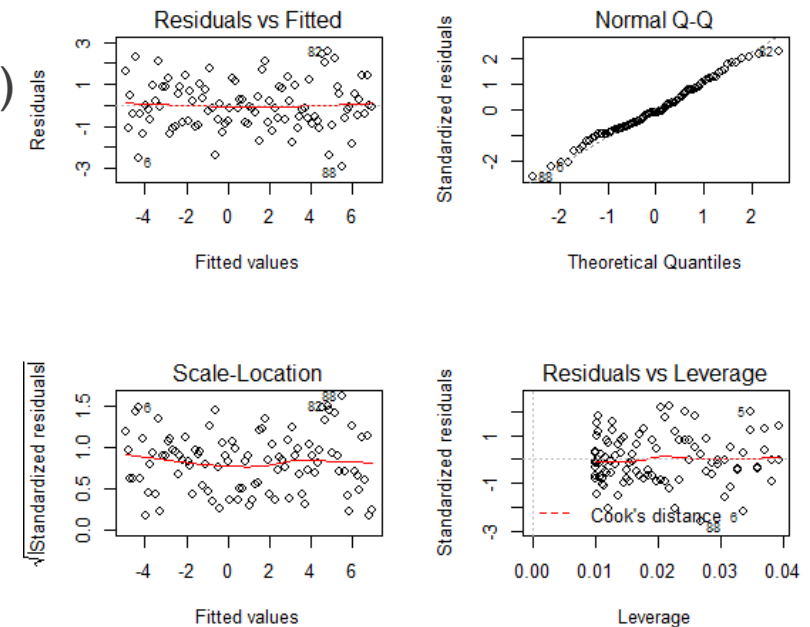
- But how well does our model predict?
- The only way to answer this question definitively is to apply the model to a *new instance of data* (test set)!
- MSE (mean squared error) characterizes how far the predicted values are from the true ones (i.e. model accuracy): $MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{f}(x_i))^2$ (so the MSE is also simply RSS/n)

```
# choose 20 points to generate new (test) data at. Might as well sample them randomly
# and independently from, say, uniform on [-3,3], it does not matter:
> xt=sample(x,20)
> yt=1+2*xt+rnorm(20,sd=1)
> yt.pred=predict(m,newdata=data.frame(x=xt))
> mse.test=sum((yt.pred-yt)^2)/20
> mse.test
[1] 1.318115
# what was the MSE on the training set? :
> rss/length(x)
[1] 1.301152
```

MODEL QUALITY

- It is important to check the overall quality of the model
 - Are the assumptions met?
 - Are there outliers in the data?
- `plot()` function is overloaded for the linear model objects (see `?plot.lm`)
 - Generates *model diagnostic plots*

```
> class(m)
[1] "lm"
> oldpar=par(mfrow=c(2,2))
> plot(m)
> par(oldpar)
```

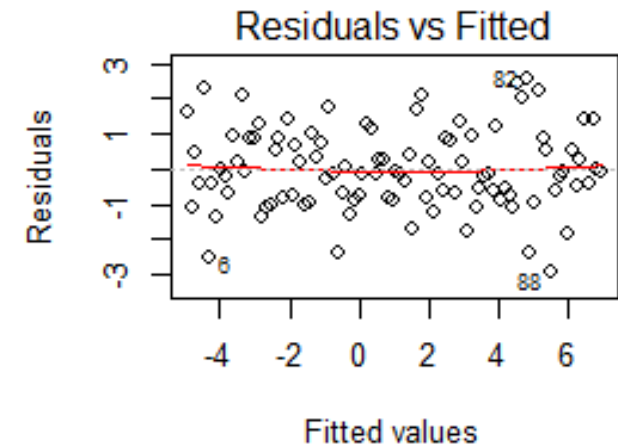


RESIDUALS VS FITTED

- The first diagnostic plot is residuals vs fitted (e.g. `plot(predict(m), y - predict(m))`)

\hat{y}_i e_i

- The simulated data we generated are nice and clean, this is how nearly ideal residuals plot should look: distribution of residuals is indeed uniform across the whole range of data values, there is no trend (straight horizontal fit)

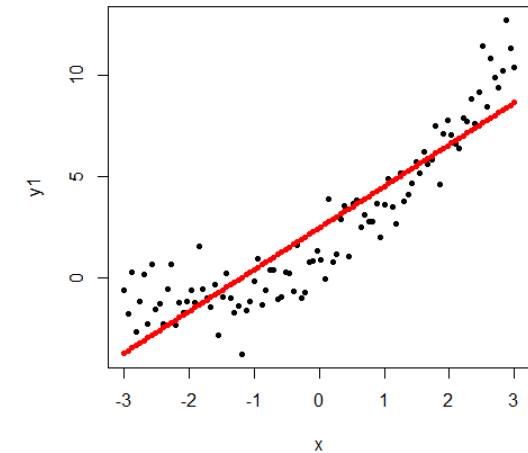


RESIDUALS VS FITTED: SIGNS OF TROUBLE

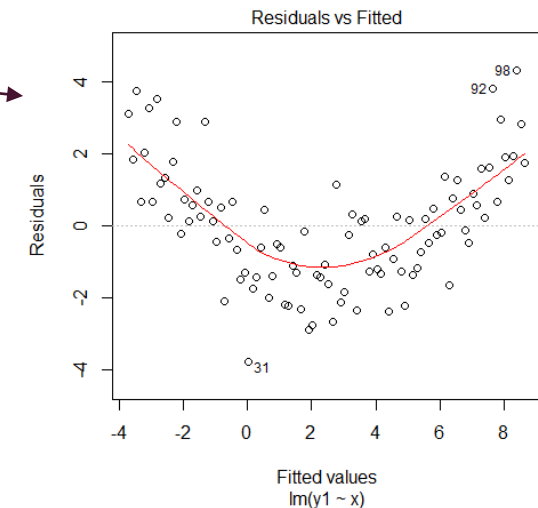
- Let us simulate a dependency that's not linear:

```
> y1=1+2*x+0.5*x^2+rnorm(100,sd=1)
> plot(x,y1,pch=19,cex=0.7)
> points(x,predict(lm(y1~x)),pch=19,cex=0.7,col="red")

> plot(lm(y1~x),which=1)
```



- We see strong indication of nonlinearity in the residuals plot:
 - Remedy: add terms to the model!

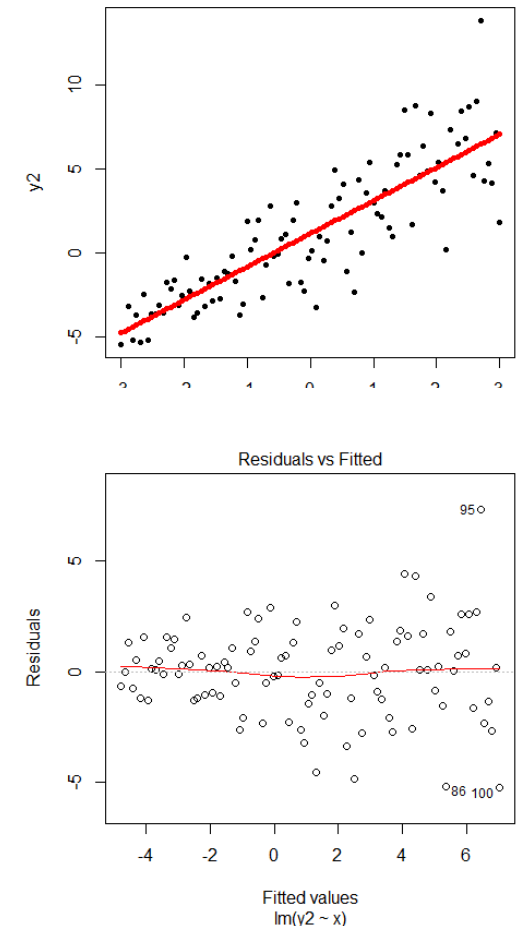


UNEQUAL VARIANCE

- Let us now simulate same linear dependence as earlier, but let the variance in the data grow with x:

```
> y2=1+2*x+rnorm(100,sd=seq(1,3,length=100))
> m2=lm(y2~x)
> plot(x,y2,pch=19,cex=0.7)
> points(x,predict(m2),pch=19,cex=0.7,col="red")
> plot(m2,which=1)
```

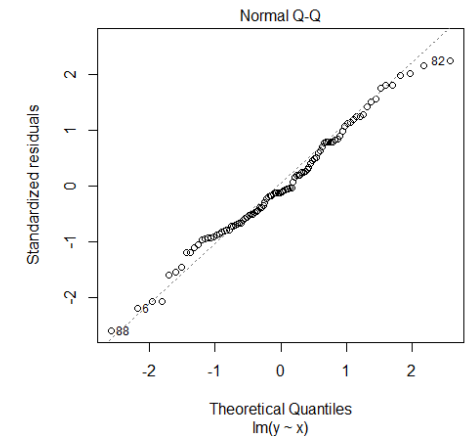
- Strong indication of heteroscedasticity: the funnel shape of the residuals
 - Note that a few points are marked as potential outliers – in our case they are not, of course, they simply come from the end with high variance, but their residuals are much larger than the *average* variance
 - Remedy: variance-stabilizing transformations ($\log Y$ or \sqrt{Y} is a decent way)



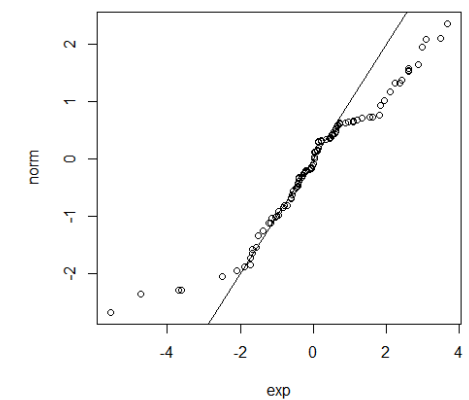
QUANTILE-QUANTILE PLOT

- In general, this is a very useful plot for (visual) comparison of the *shape* of two distributions.
- Consider two distributions
 - In our case, one is the distribution of the (observed) residuals, and the other is a reference normal distribution (think `rnorm()`)
 - In our case, the distributions are also standardized as it's the shape that we are after.
 - Draw a scatterplot of quantiles: consider for instance 7% quantile: x coordinate is the 7% quantile of the distribution 1, and the y coordinate is the 7% quantile of the distribution 2.
 - Obviously, if the distributions are the same, the QQ plot is a diagonal line
 - If, for instance, distribution 2 grows *faster* on the left or decays faster on the right than the distribution 1, then its z% quantile is reached *sooner* thus the z% quantile point will be *below* the diagonal. If the distribution 2 grows (on the left) or decays (on the right) slower, then it reaches z% quantile later than distribution 1, so the QQ plot is above the diagonal

```
> qqplot(c(rexp(50), -rexp(50)), rnorm(100), xlab="exp", ylab="norm")  
> abline(0,1)
```



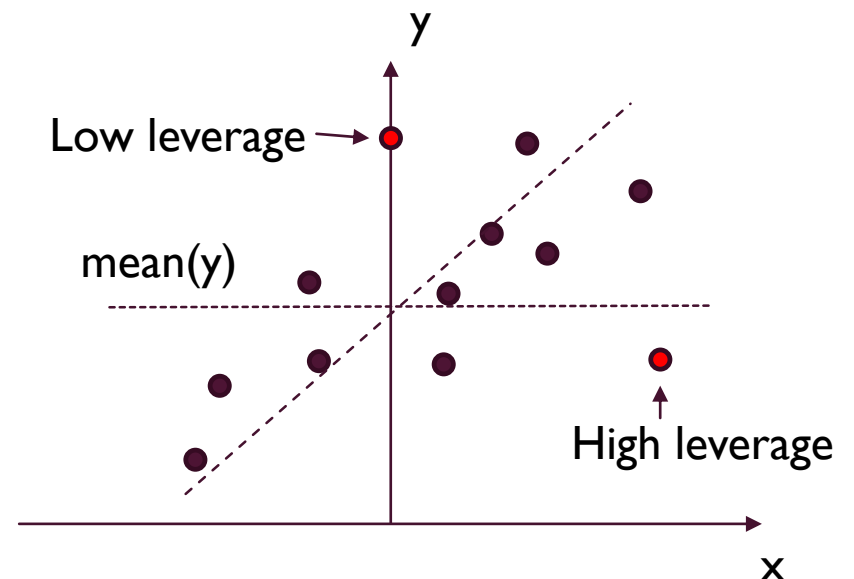
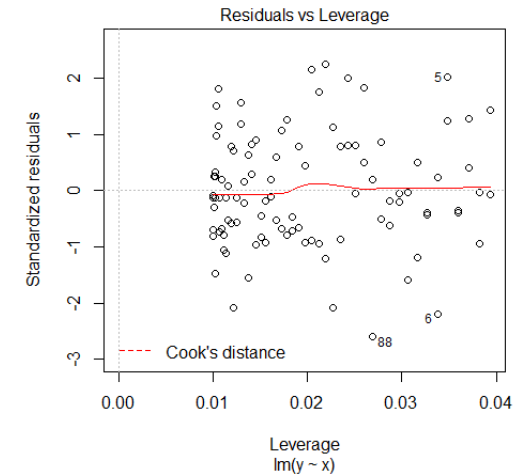
Our model



Example: normal
vs symmetric exp

HIGH LEVERAGE OBSERVATIONS

- Think of it as data points literally pushing on the regression line as a lever.
 - The farther they are along the x axis from $\text{mean}(x)$, the longer the lever (=high leverage!)
 - The farther they are along the y (=large residual), the stronger they are pushing
 - The residuals vs leverage plot show just that: the leverage and residual of each datapoint. The points with high leverage *and* high residual value are potential outliers that affect your fit strongly and contribute unfairly much to it. Points with large residual but low leverage are possibly outliers too, but they don't affect the fit much!



CATEGORICAL PREDICTORS

- In the framework of linear regression, categorical variables are no different from continuous ones
 - For a two-level variable, we can simply encode one level as 0 and another as 1
 - What would the formula $y = \beta_0 + \beta_1 x + \varepsilon$ signify then? What do the model coefficients mean (think about $x=0$ and $x=1$)
 - For variables with more than two levels we *cannot* represent those as e.g. 0, 1, 2... (why?)
 - Instead we should introduce separate indicator variables 0/1, the combination of which would encode different levels

```
> boxplot(Sepal.Length~Species,data=iris,col=c("lightgreen","lightblue","magenta"))
> summary(lm(Sepal.Length~Species,data=iris))
...
```

Coefficients:

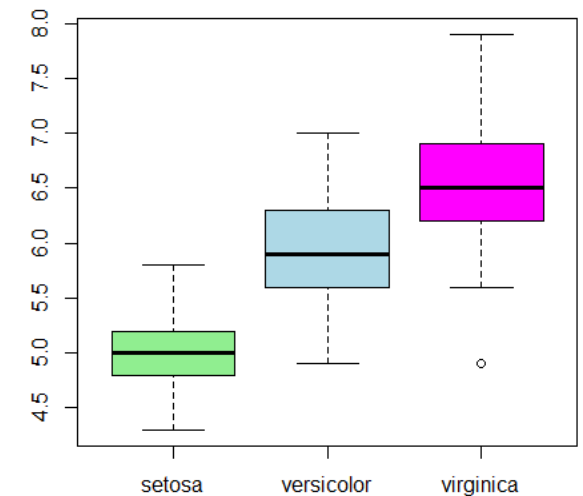
	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	5.0060	0.0728	68.762	< 2e-16 ***
Speciesversicolor	0.9300	0.1030	9.033	8.77e-16 ***
Speciesvirginica	1.5820	0.1030	15.366	< 2e-16 ***

...

```
> mean(iris$Sepal.Length[iris$Species=="setosa"])
[1] 5.006
> mean(iris$Sepal.Length[iris$Species=="versicolor"])-
mean(iris$Sepal.Length[iris$Species=="setosa"])
[1] 0.93
```

CSCI-E63C ELEMENTS OF STATISTICAL LEARNING

Note that β_0 ="average at the base factor level" is just the default choice in R (but not the only one possible; outside of our scope, see 'contrasts')



SUMMARY

- We have discussed MLE and observed how the estimator for the distribution location (mean) and for the linear regression coefficients (intercept and slope) naturally arise as optimal parameters that maximize the likelihood (i.e. ensure that the probability to sample the data as observed is the largest), *under the assumption of normal noise distribution*.
- We further revisited the concept of statistical (in)dependence and considered the simple case of two (linearly) dependent variables (plus noise!)
- We learned how to perform simple linear regression in R
- We discussed accuracy of the linear regression coefficients (what's the uncertainty of the values we calculate from the noisy data), and their significance (how confident we are that the effect we observe, e.g. non-zero slope, is *real* rather than the result of a random noise fluctuation in the data).
- We further discussed the accuracy of *the model itself* (we can have a non-zero and very significant slope, but the predictions we are making are still poor: that means that there is lots of noise in the data and/or the dependence is not a linear one – ie. even the best linear model we can possibly get is still grossly inadequate! We have also briefly discussed the difference between model accuracy on the training and test datasets.
- Finally, we looked at model diagnostics (are the assumptions of the model met? Do the data contain outliers?)