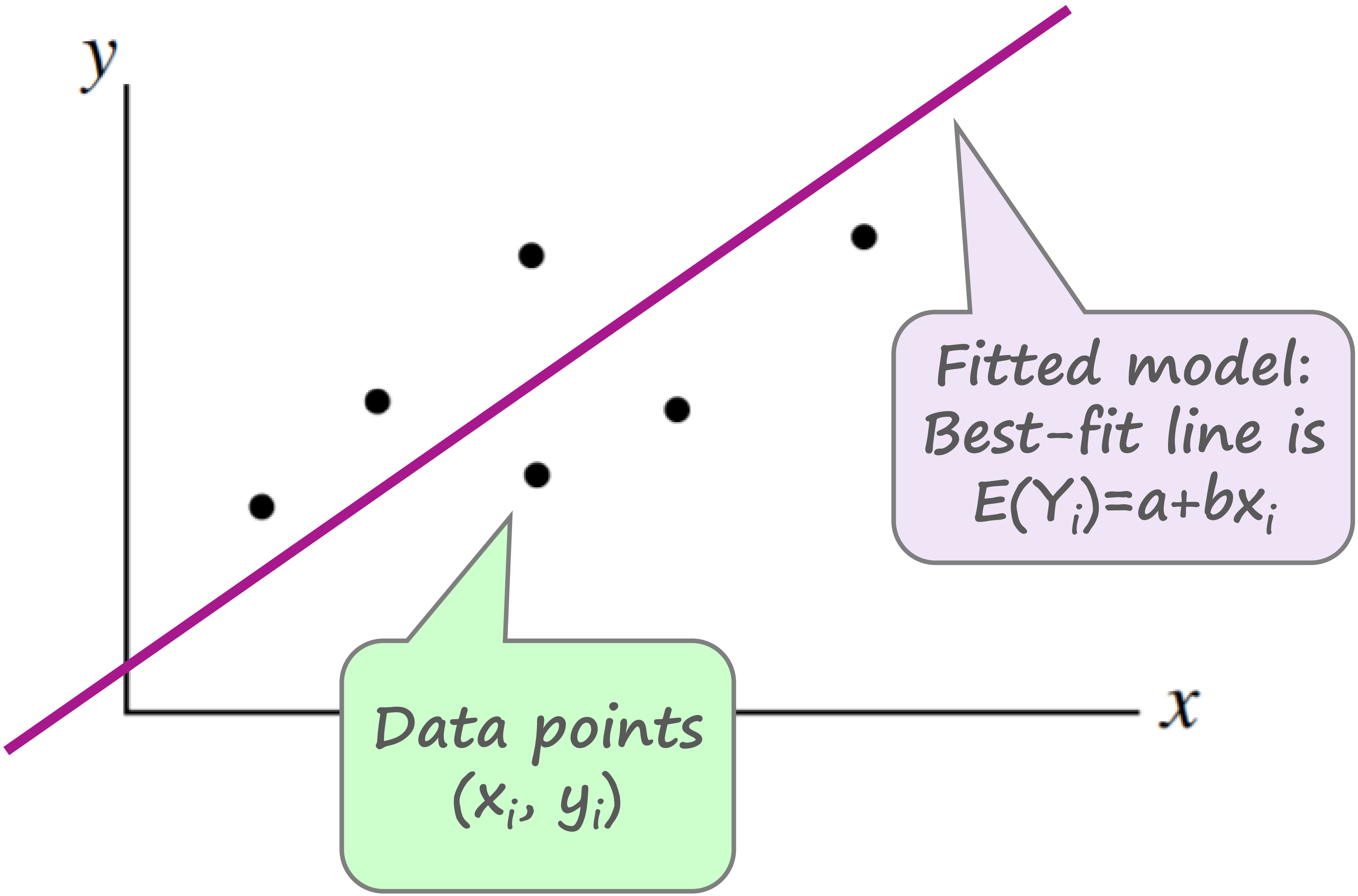
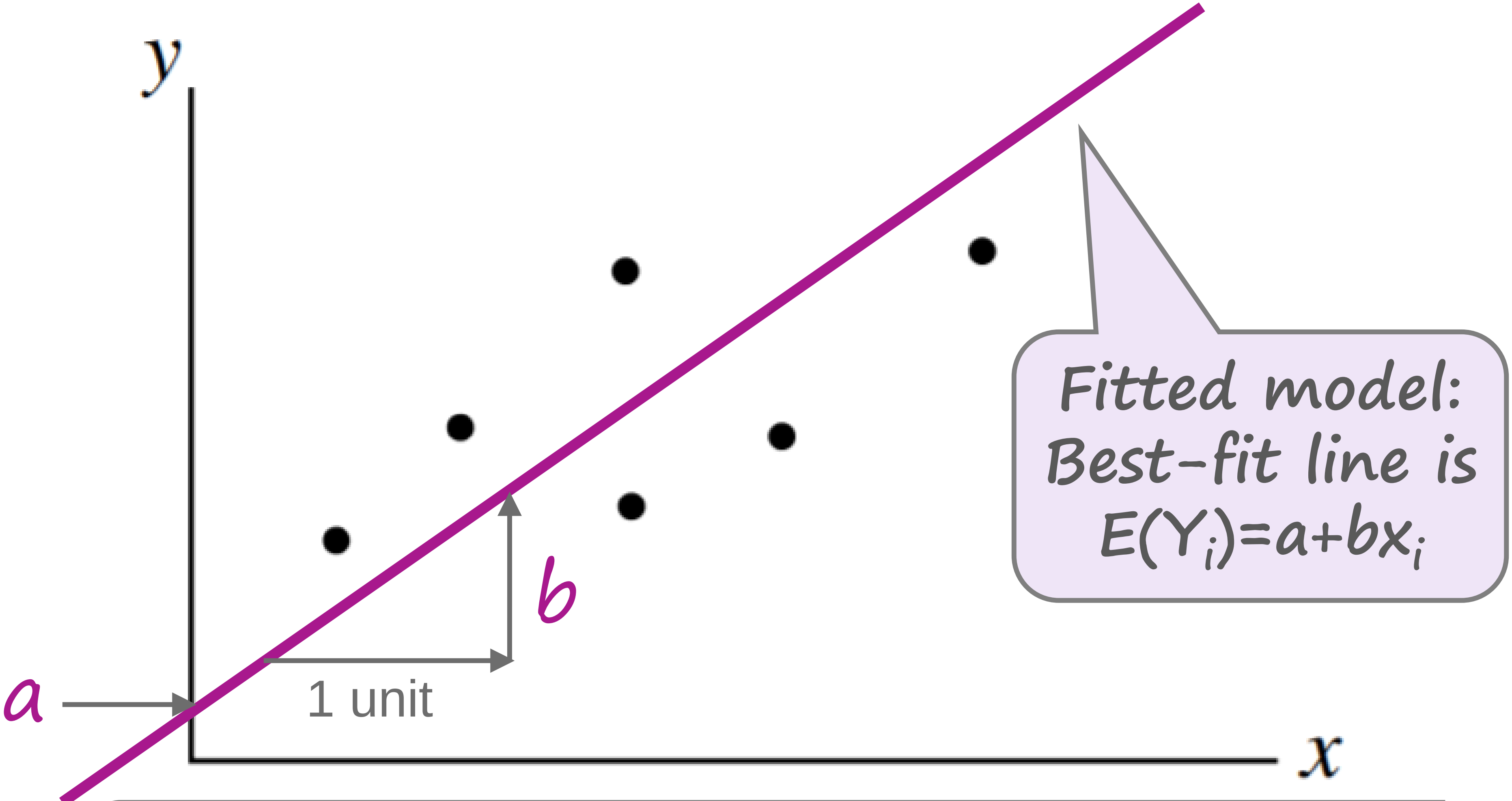
DIY Statistical Modelling

Demystifying the black box …

How do we fit a model?

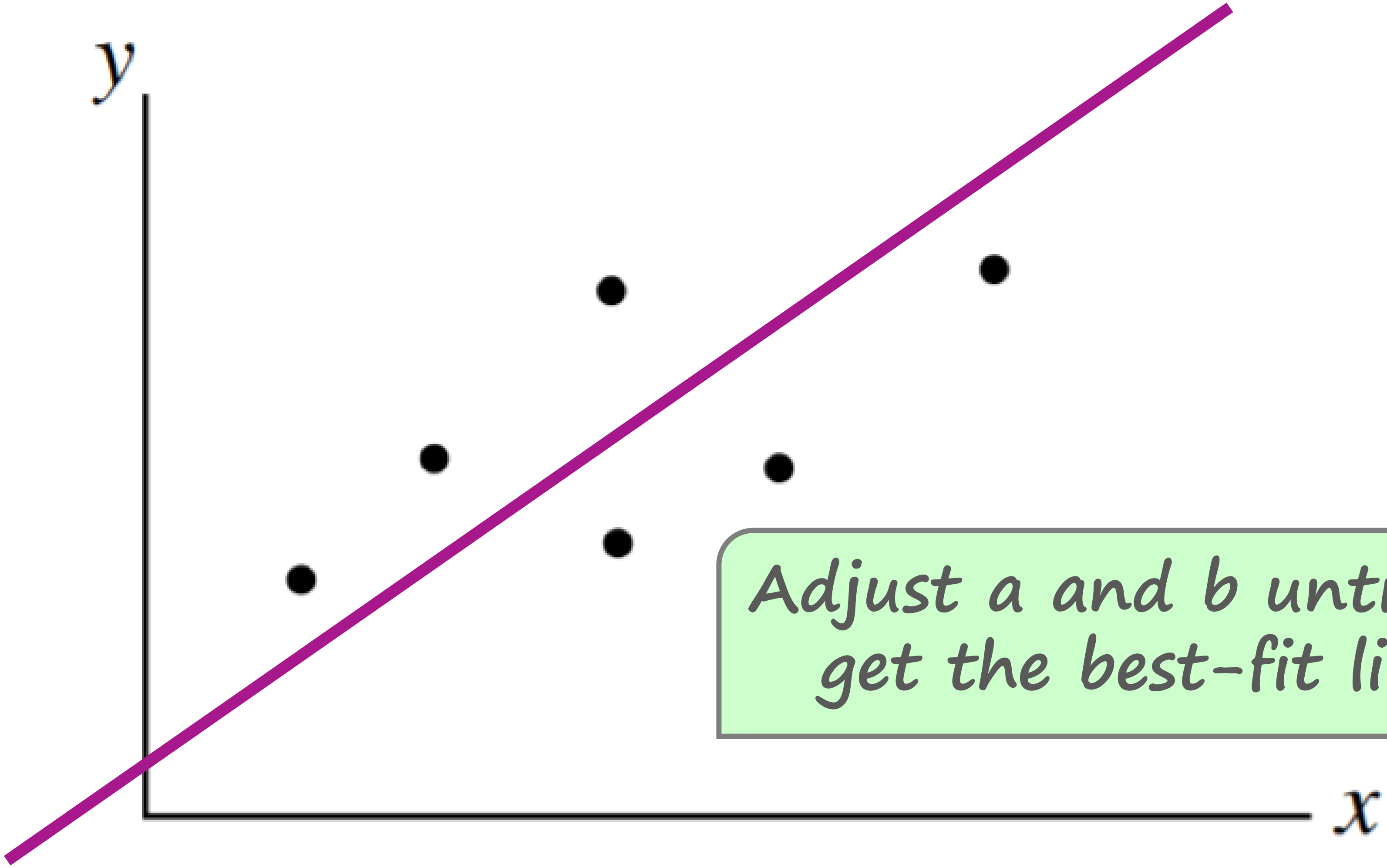


How do we fit a model?



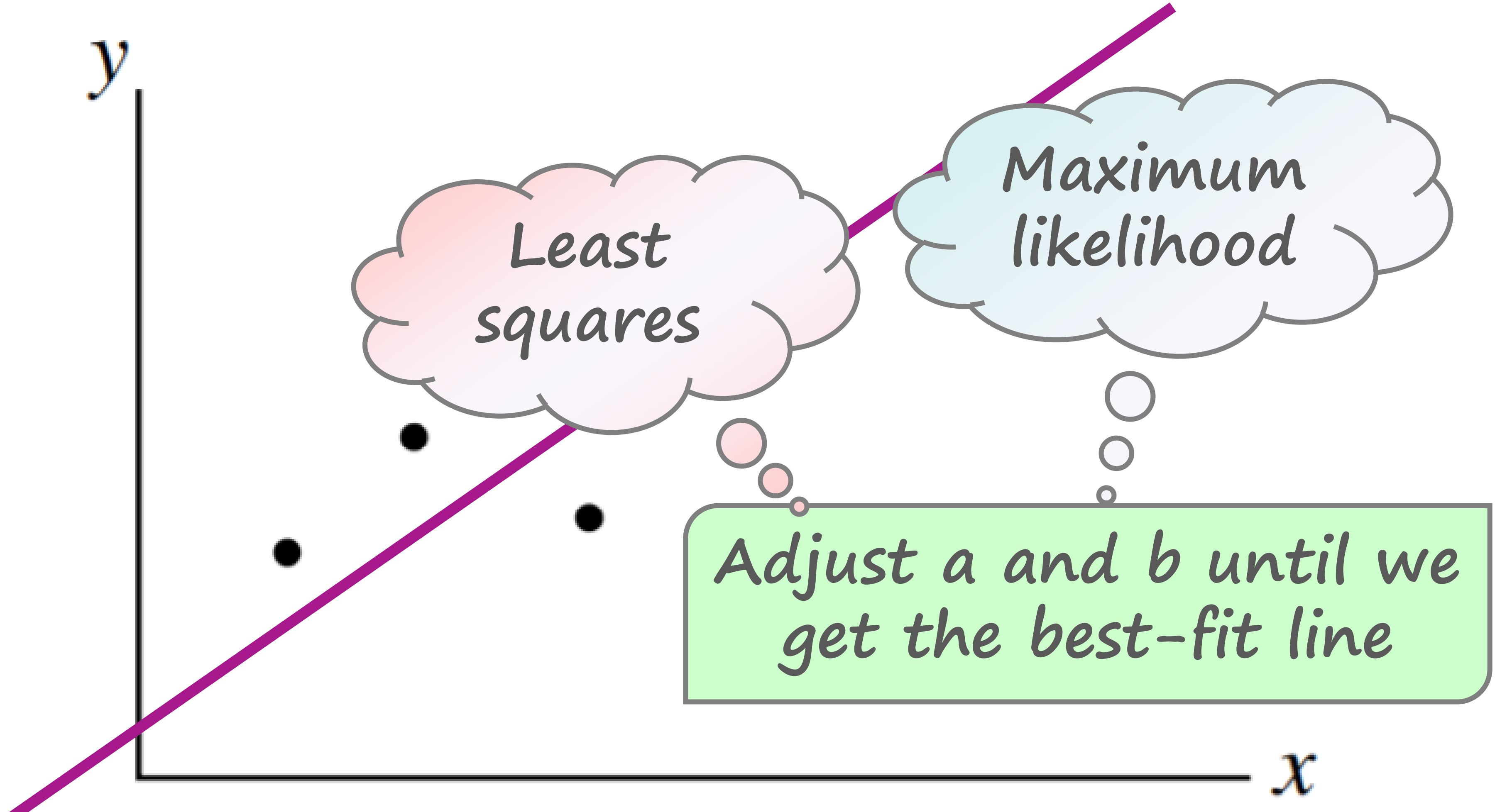
Fitting the model means estimating values for the unknown parameters, a and b

How do we estimate *a* and *b* ?



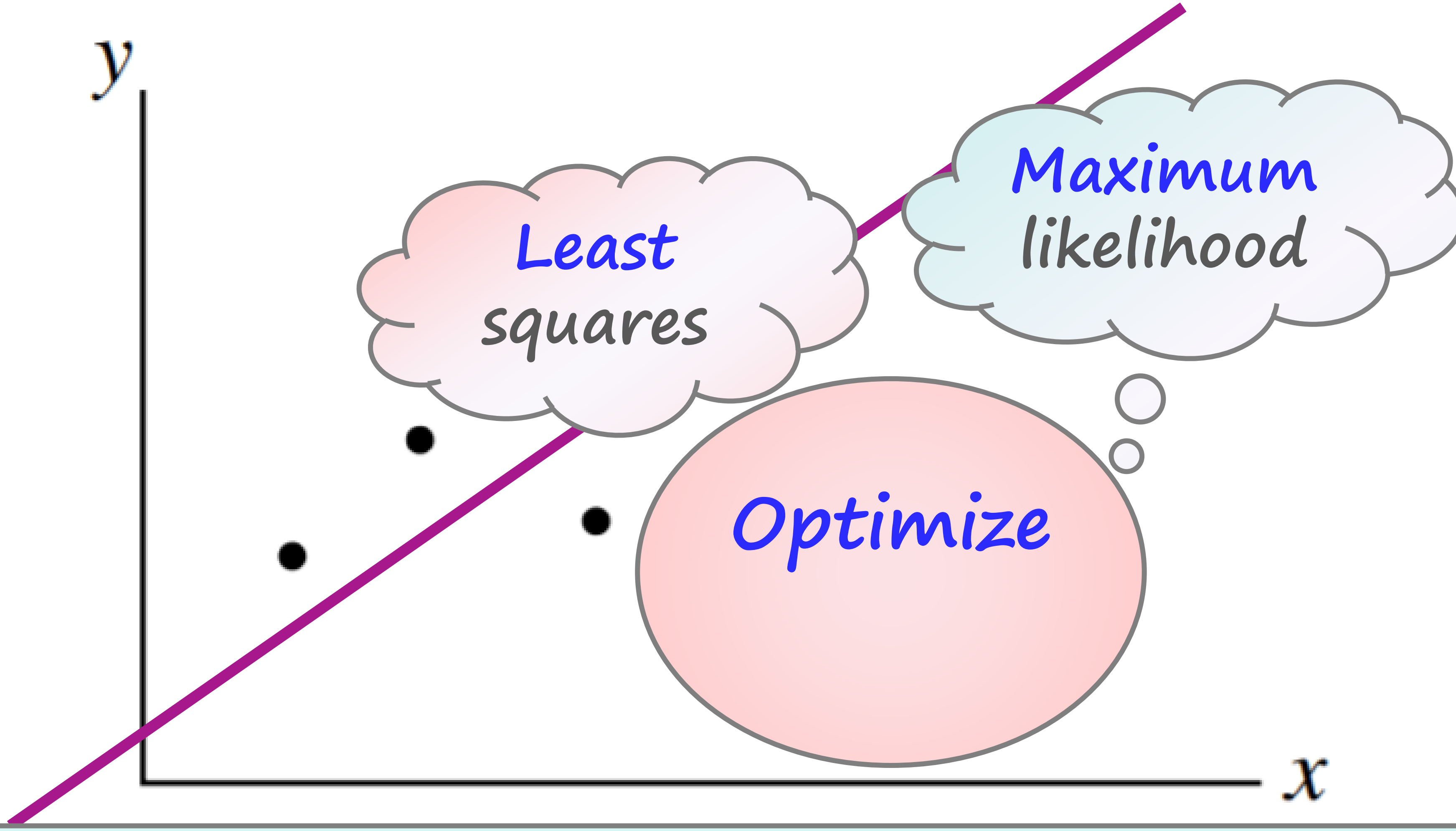
Adjust a and b until we get the best-fit line

What do we mean by ‘best fit’?



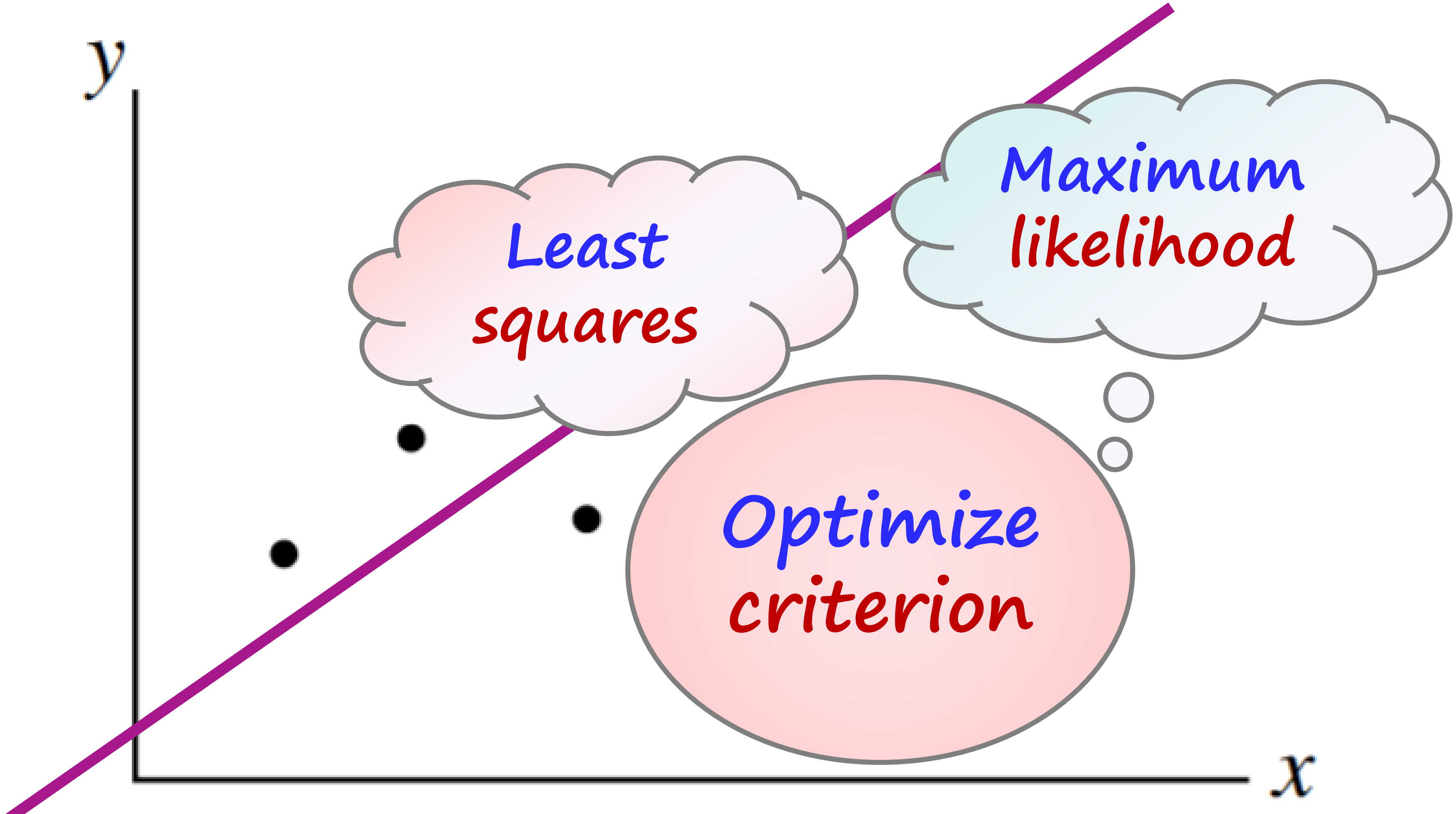
We need a criterion to optimize:
something we can measure that can become best

What do we mean by 'best fit'?



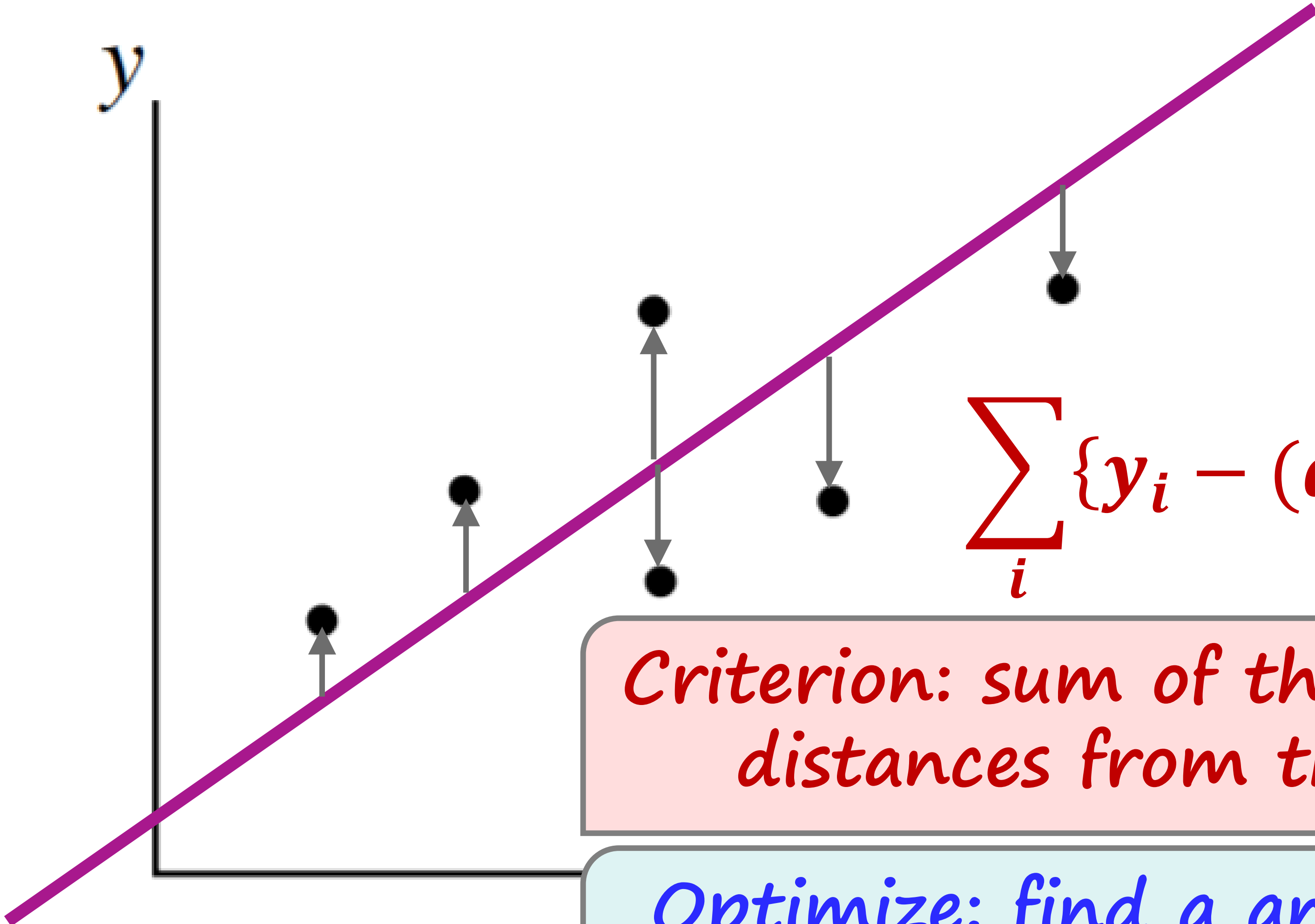
We need a criterion to optimize:
something we can measure that can become best

What do we mean by 'best fit'?



We need a criterion to optimize:
something we can measure that can become best

Least squares

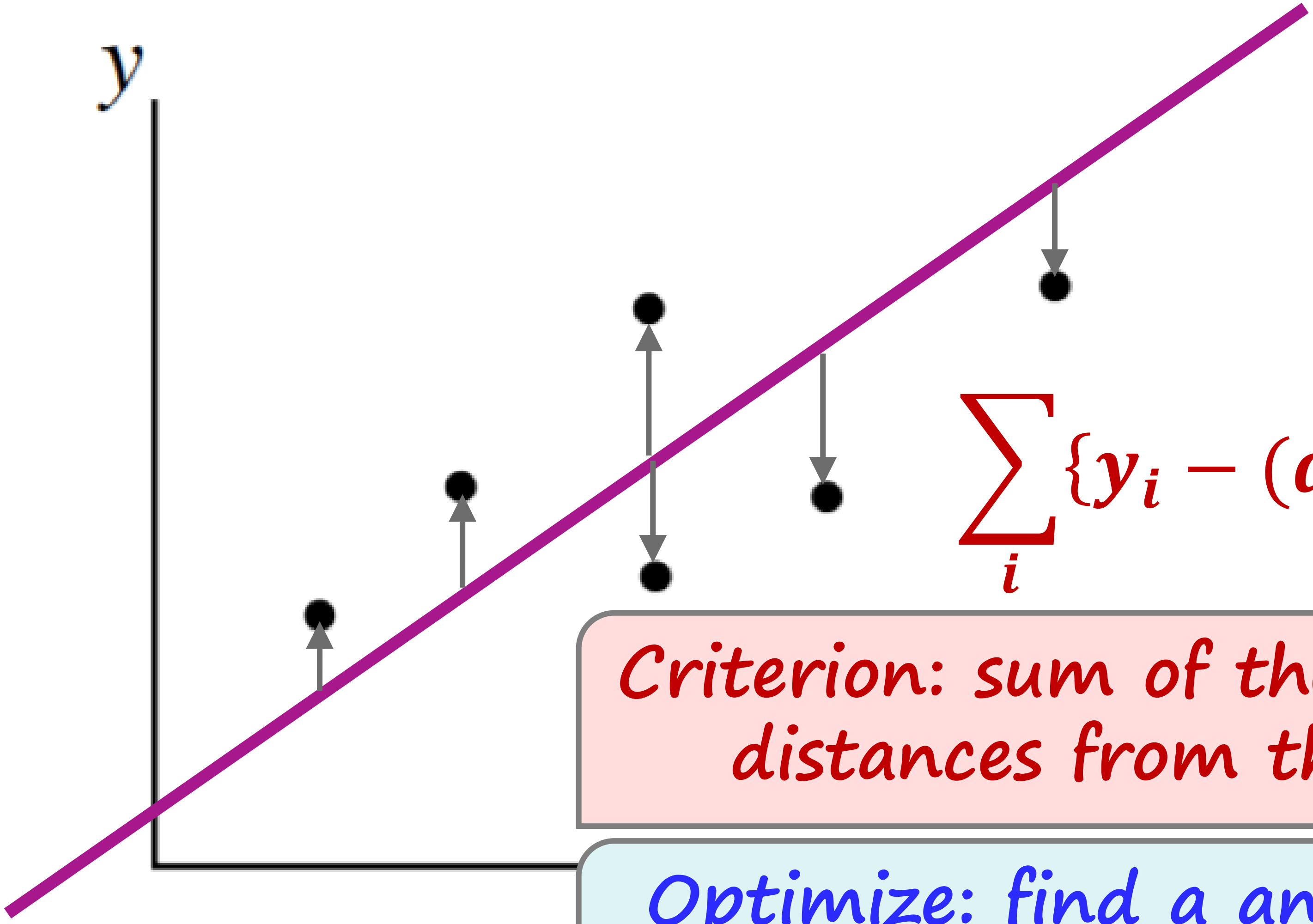


$$\sum_i \{y_i - (a + bx_i)\}^2$$

Criterion: sum of the squared distances from the line

Optimize: find a and b that minimize this sum of squares

Will we need anything else?

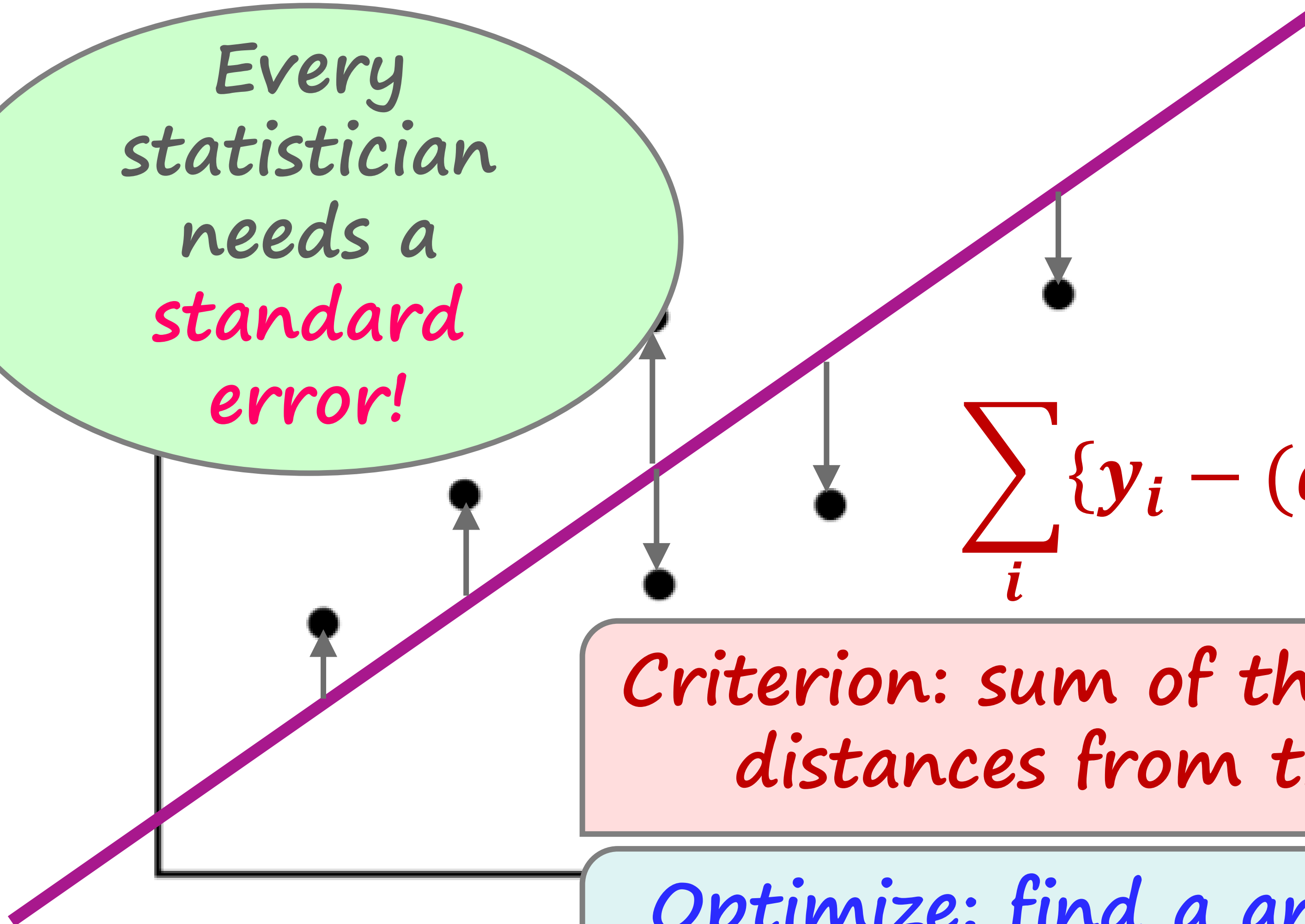


$$\sum_i \{\boldsymbol{y_i} - (\boldsymbol{a} + \boldsymbol{b}\boldsymbol{x_i})\}^2$$

Criterion: sum of the squared distances from the line

Optimize: find a and b that minimize this sum of squares

Will we need anything else?



Every statistician needs a **standard error!**

$$\sum_i \{y_i - (a + bx_i)\}^2$$

Criterion: sum of the squared distances from the line

Optimize: find a and b that minimize this sum of squares

Why do we need a standard error?

An estimate is useless without a standard error!

Would you eat this...?

This...?

Yum! (not)

???

Why do we need a standard error?

An estimate is useless without a standard error!

It's useless to try selling a possibly-dodgy item unless you include an assurance of goodness

Why do we need a standard error?

An estimate is useless without a standard error!

- **A standard error is quality assurance**

This item is
only a little bit
lethal!

Why do we need a standard error?

An estimate is useless without a standard error!

- **A standard error is quality assurance**

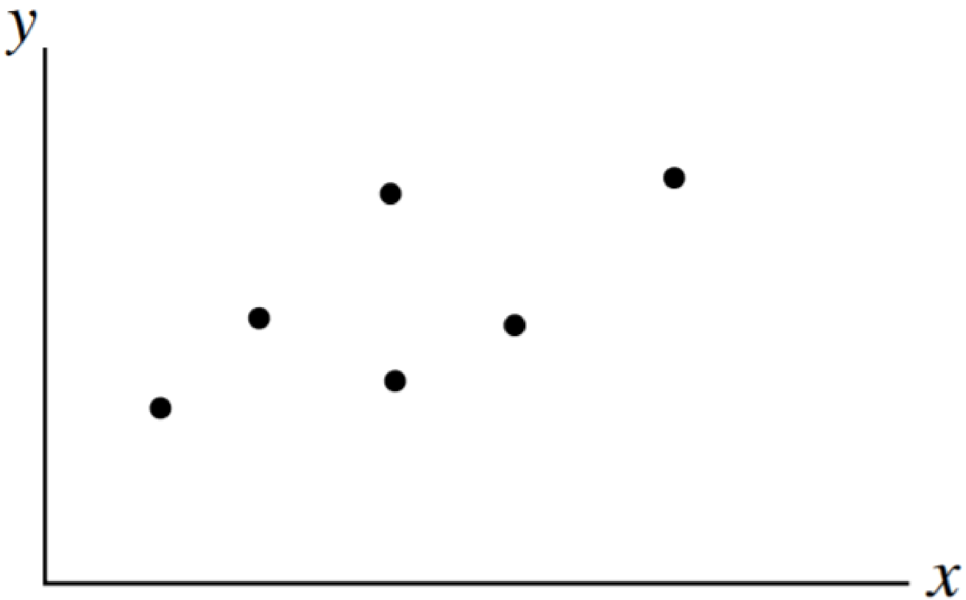
- **Anyone can propose an estimate:**

$a = 3.14$ $b = 1.23$

My estimates!

- **But we need the standard error to tell us what this estimate is worth, or how safe it is**

SE: pathway to inference (conclusions)

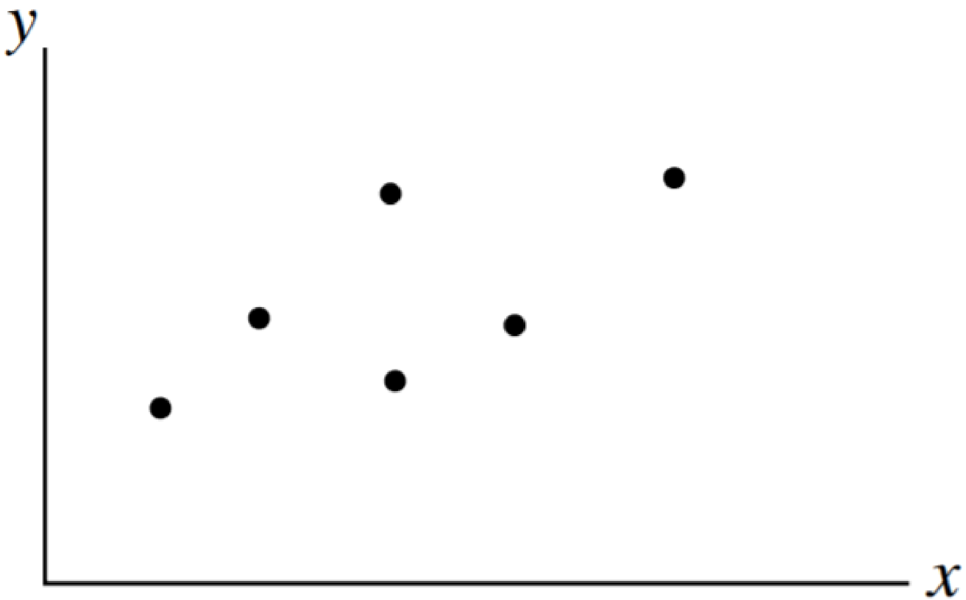


Is there a relationship between x and y?

Are we sure the slope b isn't zero?

- My estimate is b = 1.23, which isn't zero…
- But to answer the question "are we sure?", we need the standard error

SE: pathway to inference (conclusions)



Is there a relationship between x and y?

Are we sure the slope b isn't zero?

If my estimate is $b = 1.23$ …

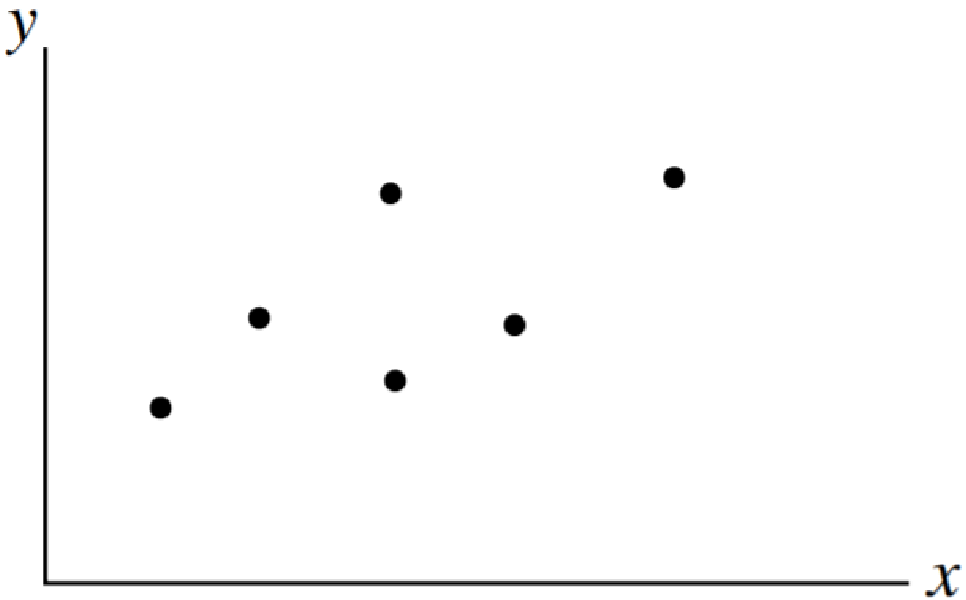
SE = 5.8

Poisonous!!
Don't trust this b!

SE = 0.2

Great stuff!
You can trust that b>0

SE: pathway to inference *(conclusions)*



Is there a relationship between x and y?

↓

Are we sure the slope b isn't zero?

Standard error → p-values and confidence intervals → Conclusions

So how can we do all this ourselves?

We need:

- **A criterion to optimize**
  - ➢ **e.g. least squares or maximu**

Next video!

- **Some way to optimize things**
  - ➢ **In R, if you can calculate it, you can optimize it**

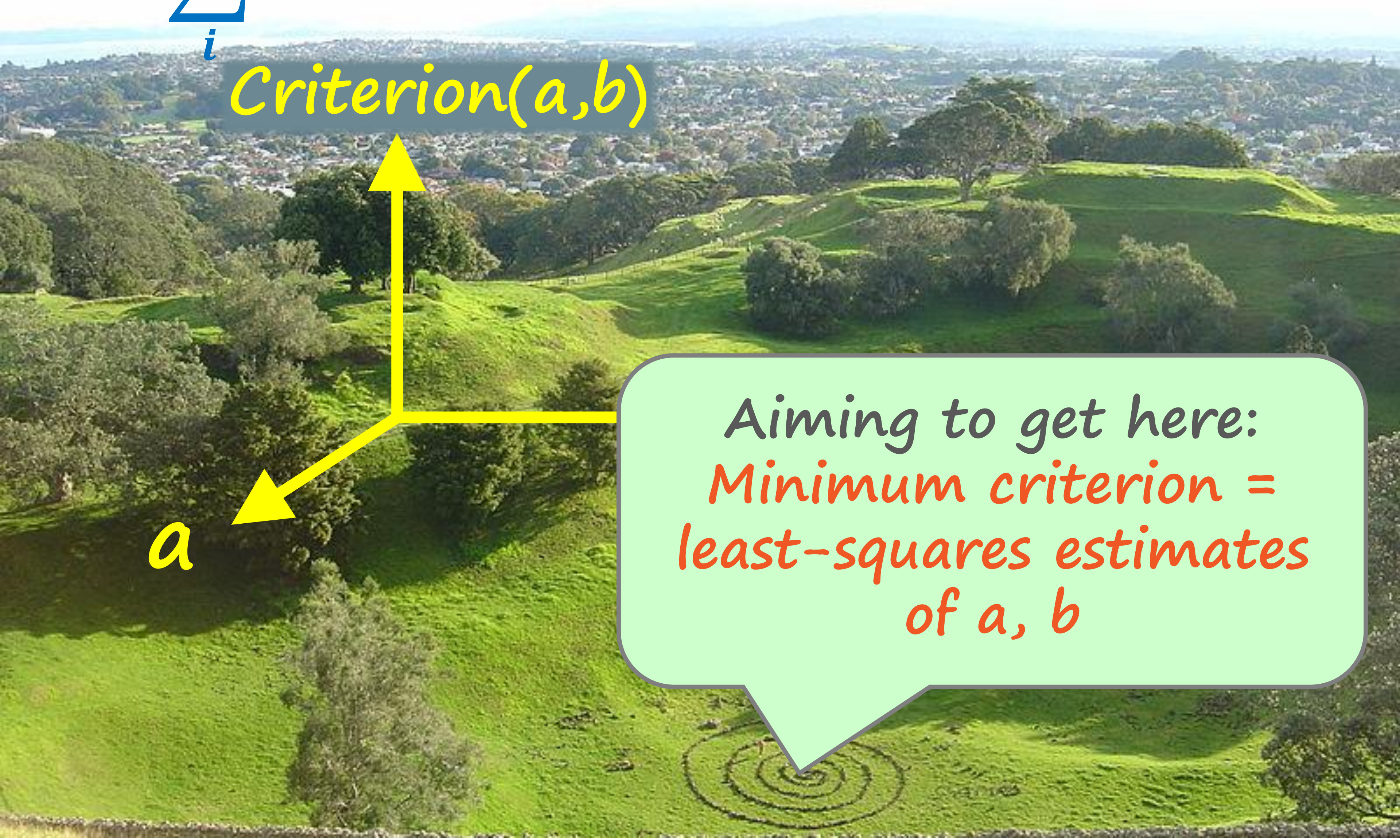
- **Some way to generate a standard error**
  - ➢ **We can use the bootstrap**

Numerical optimization in R

- In R, if you can calculate it, you can optimize it
- Numerical optimizers repeatedly try out values of a and b to find their way downhill to the minimum

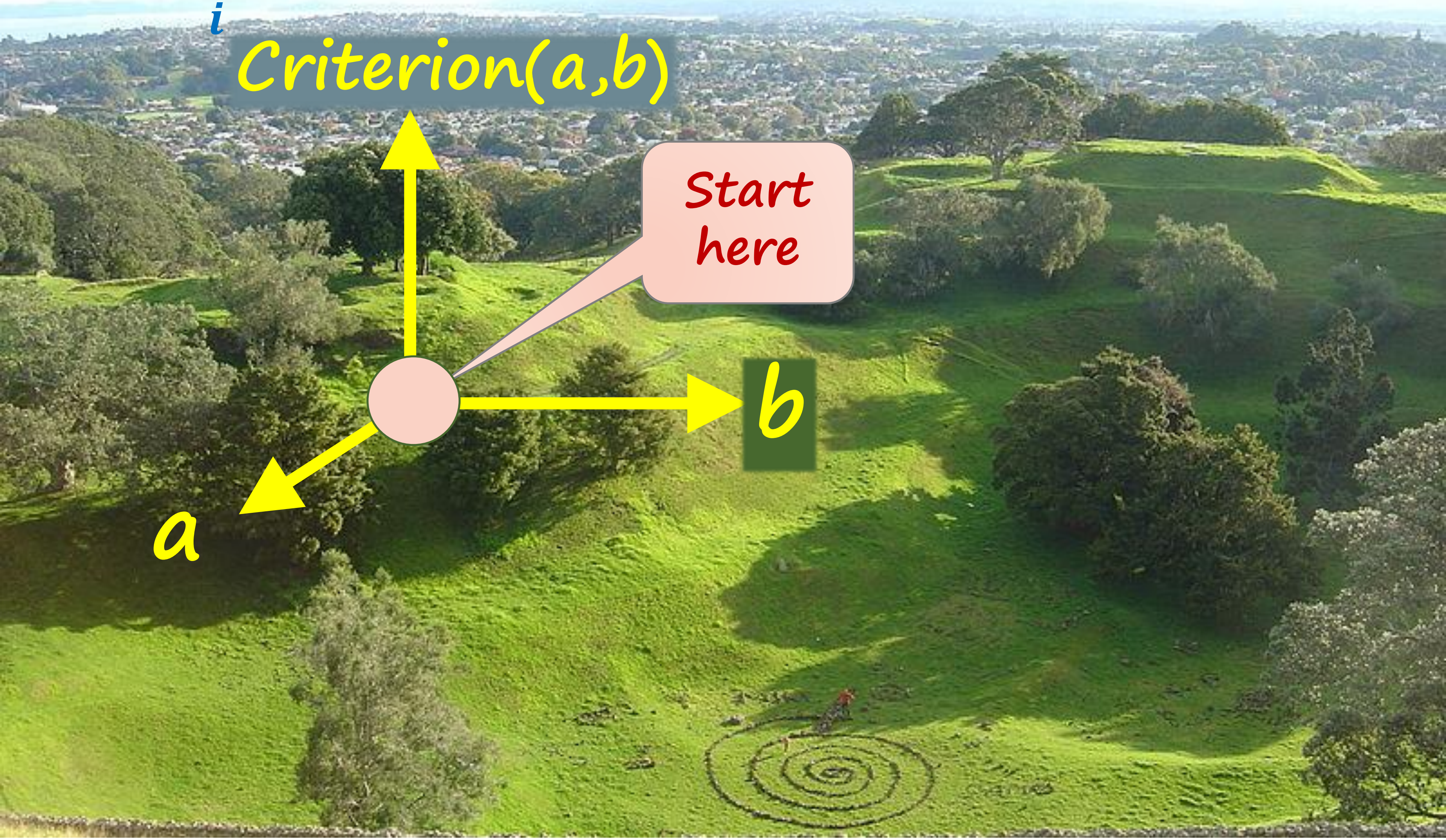

© aucklanddailyphoto.com

$$\sum_i \{y_i - (a + bx_i)\}^2$$



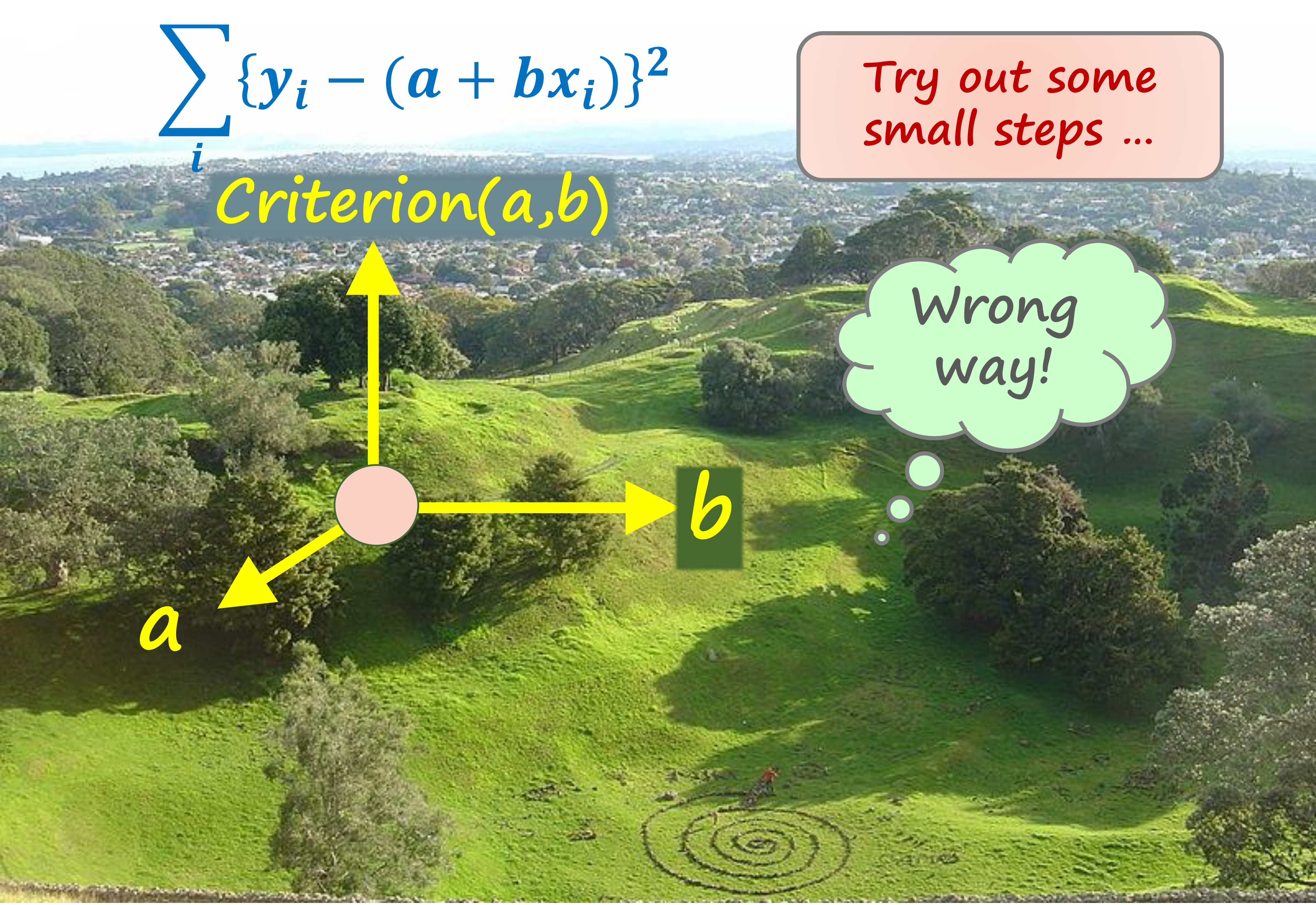
We want to fumble our way downhill to the bottom

$$\sum_i \{y_i - (a + bx_i)\}^2$$

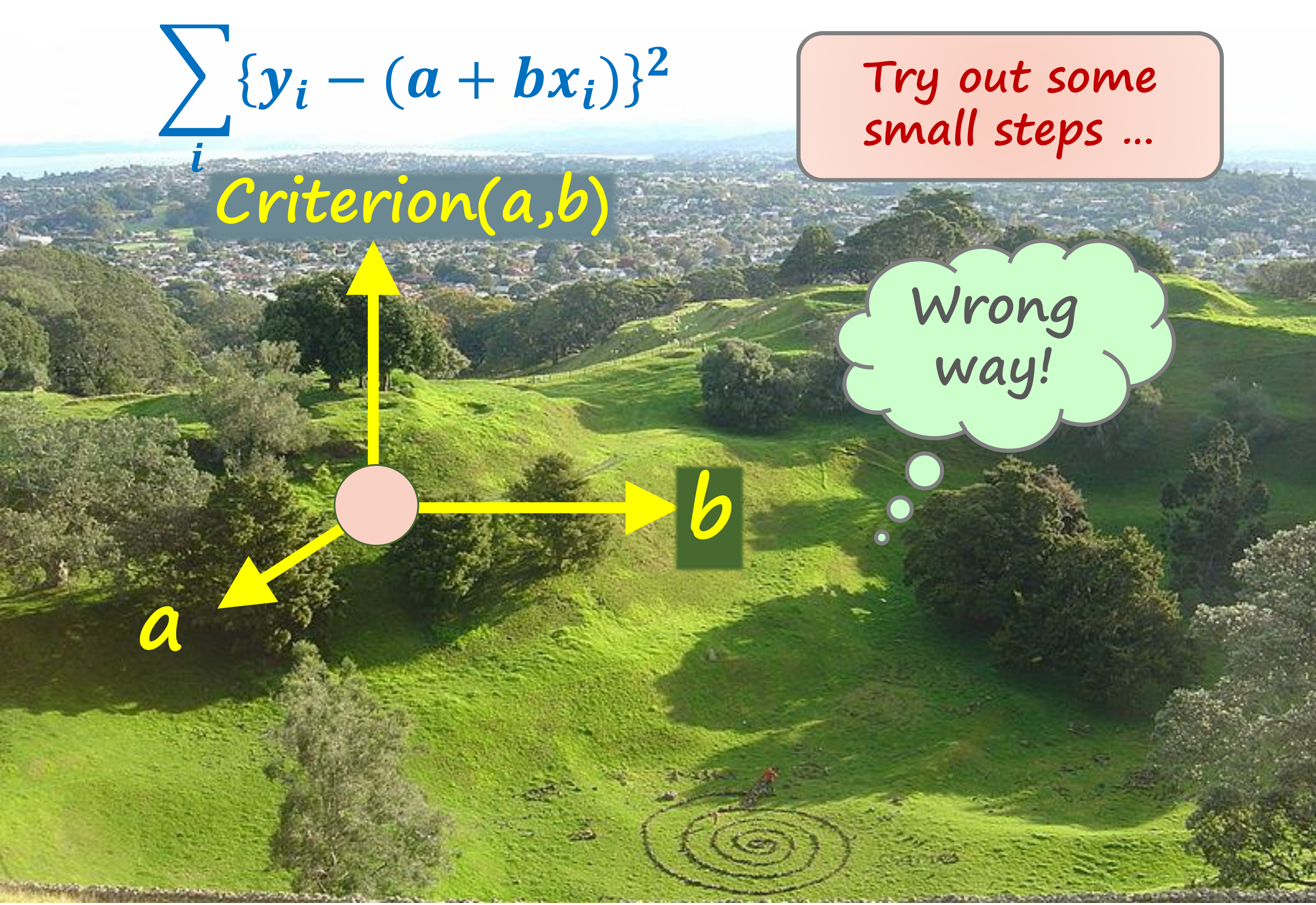


We want to fumble our way downhill to the bottom

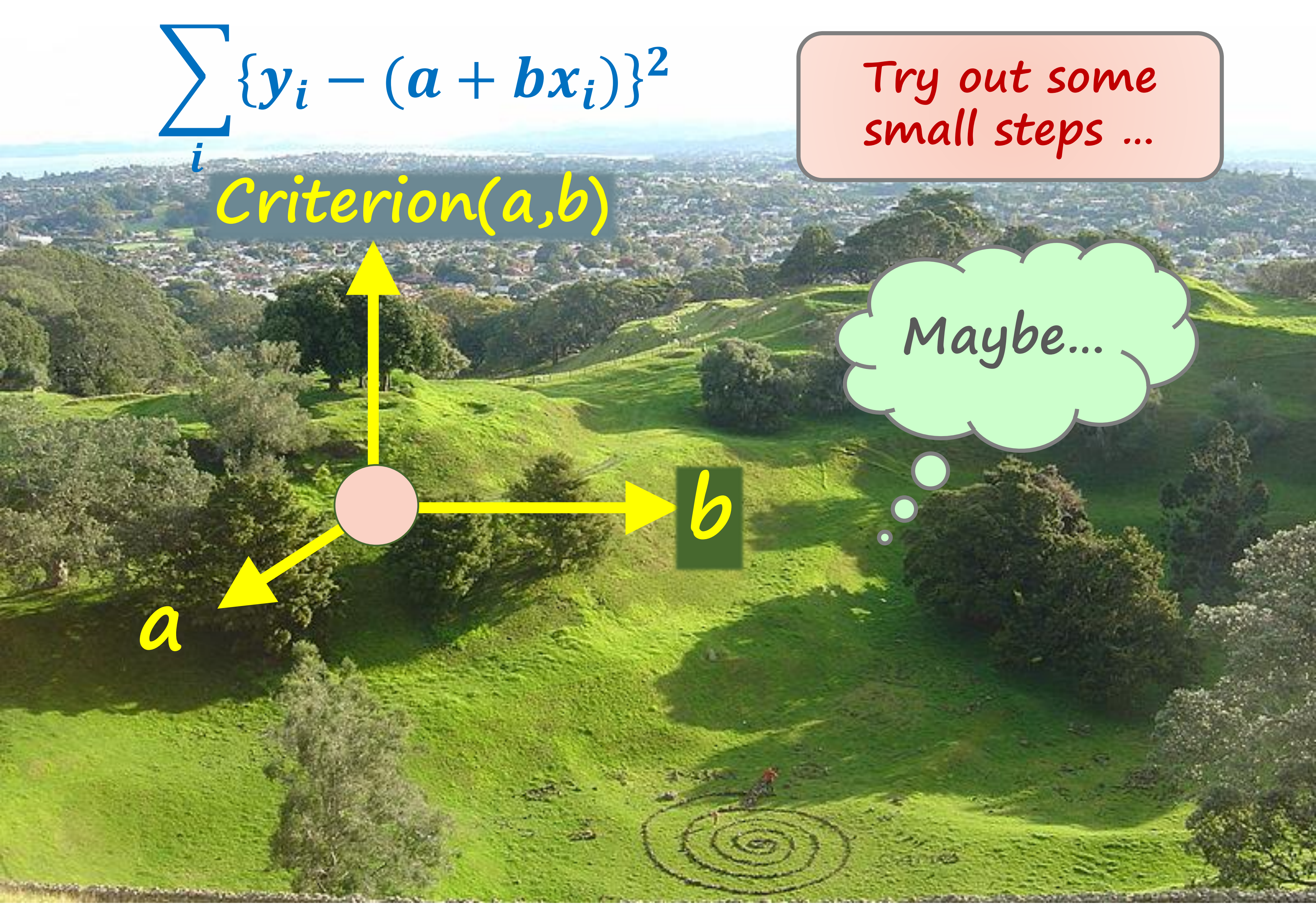
$$\sum_i \{y_i - (a + bx_i)\}^2$$



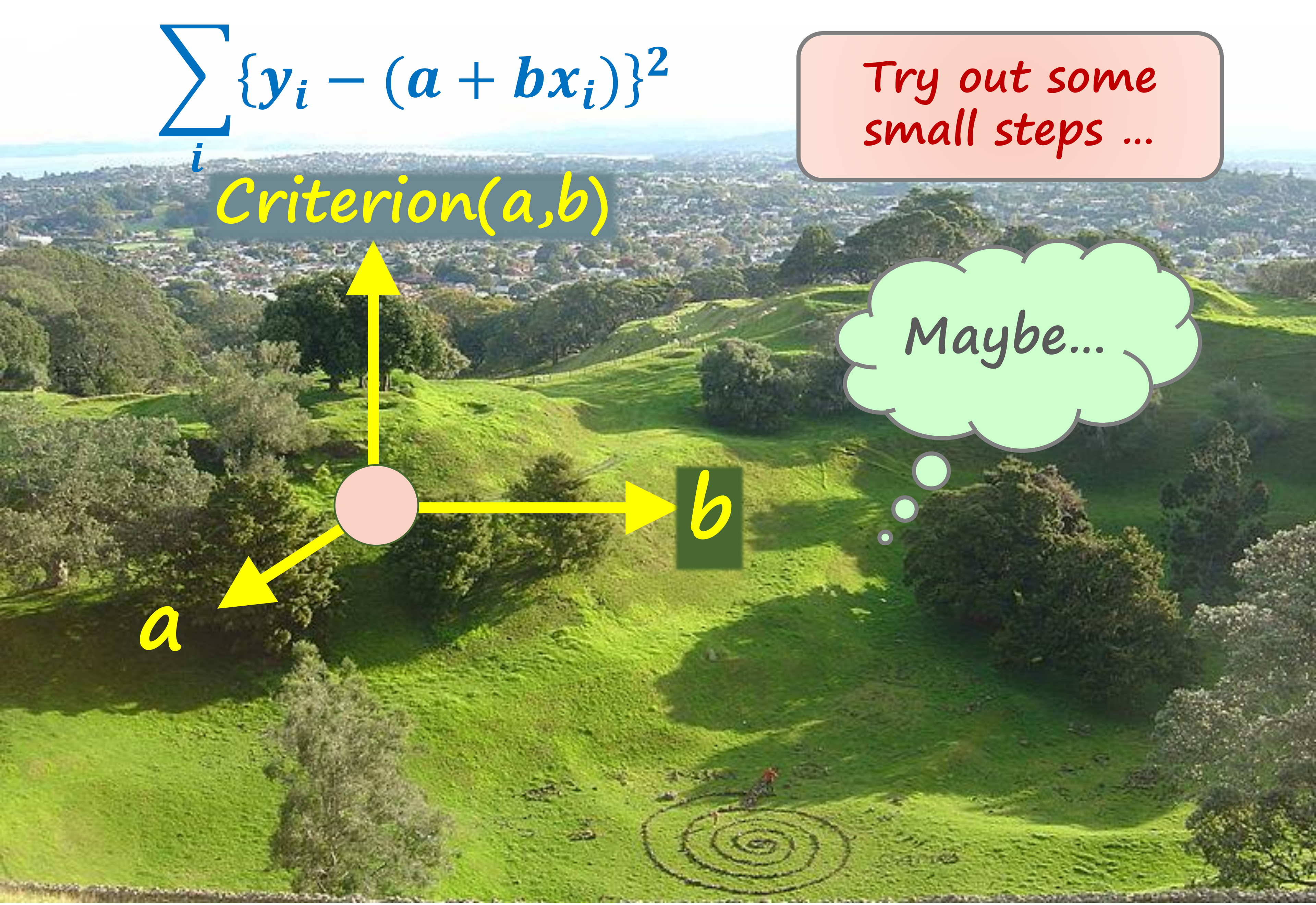
We want to fumble our way downhill to the bottom



We want to fumble our way downhill to the bottom

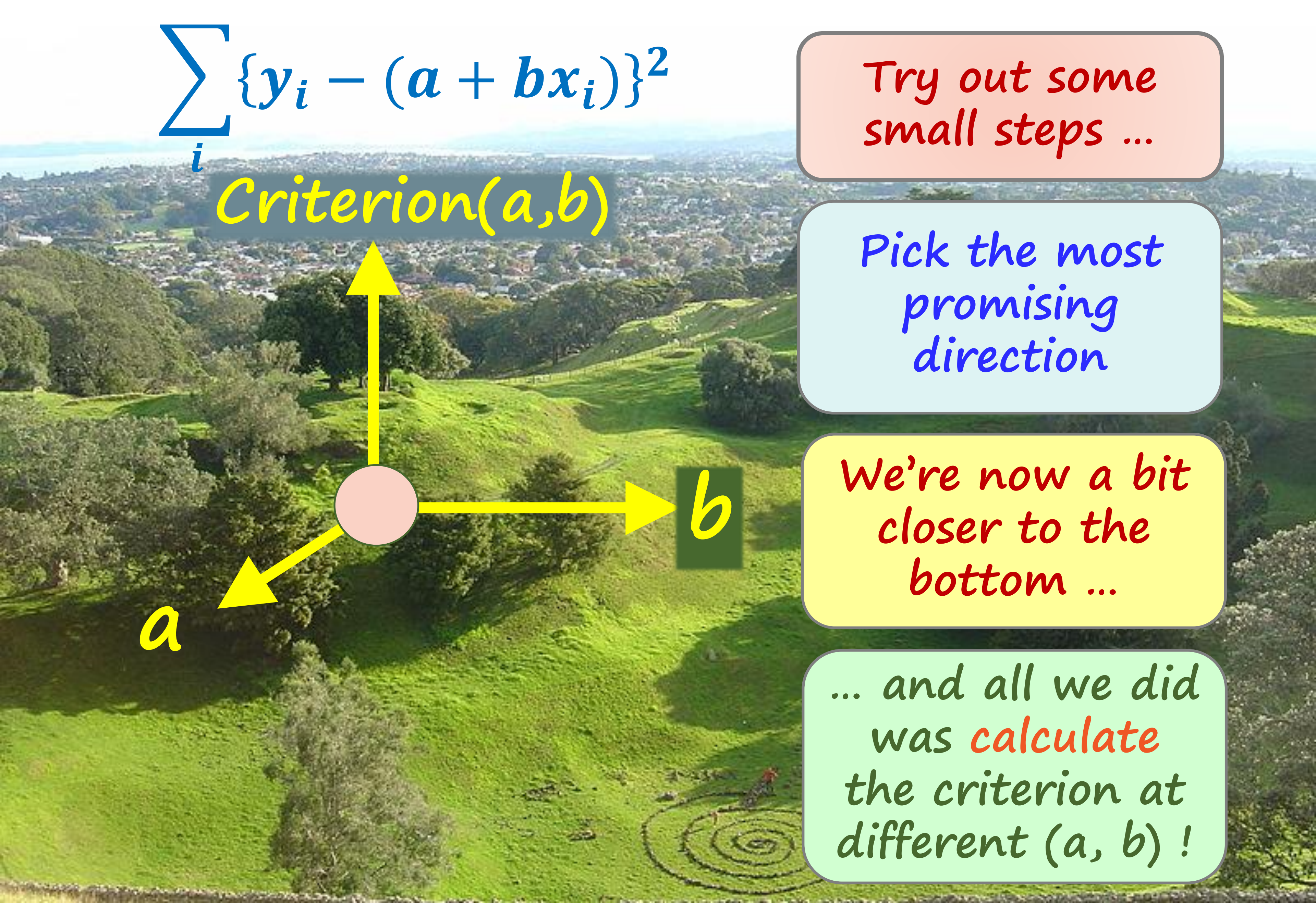


We want to fumble our way downhill to the bottom



We want to fumble our way downhill to the bottom

$$\sum_i \{y_i - (a + bx_i)\}^2$$



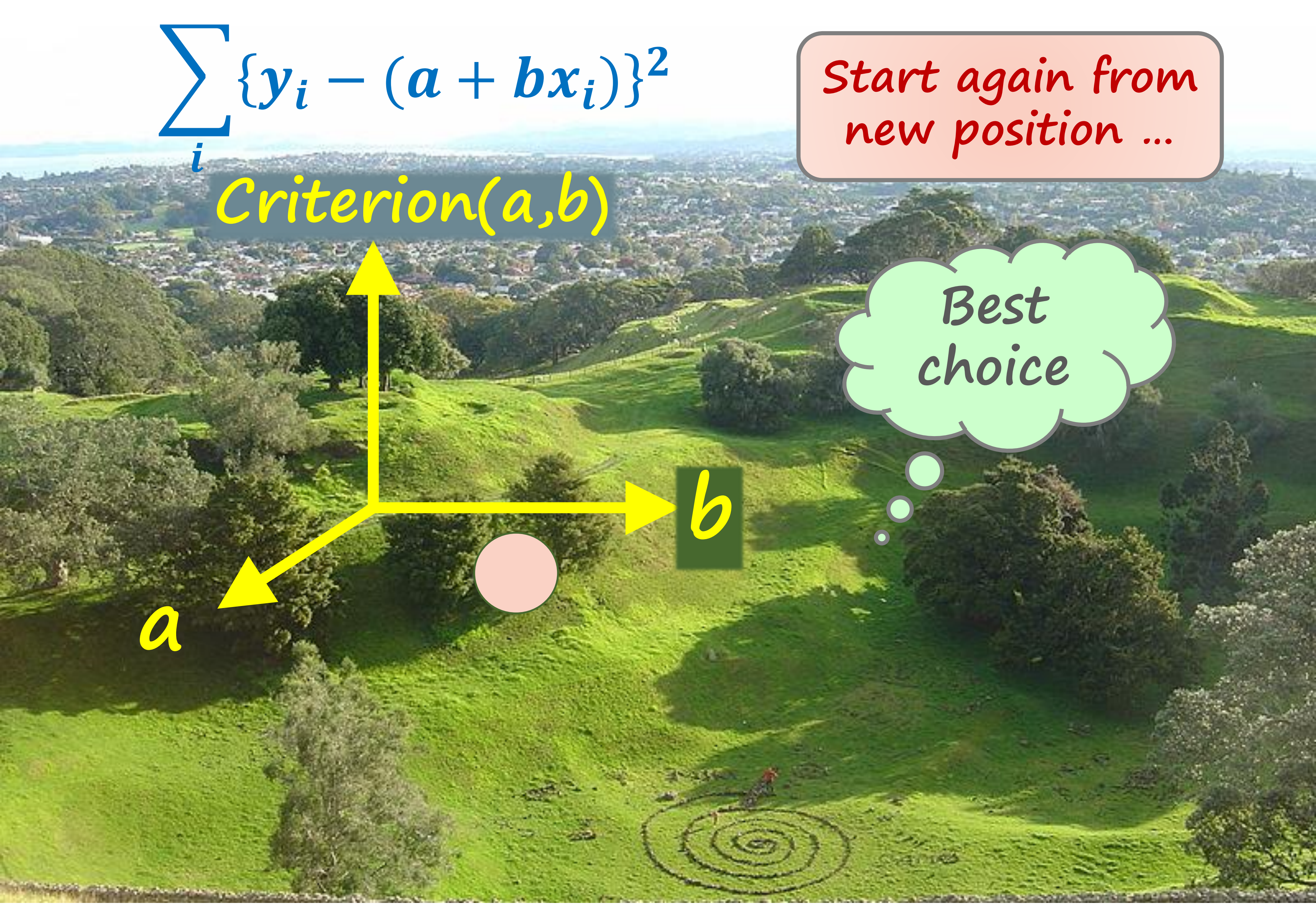
Try out some small steps …

Pick the most promising direction

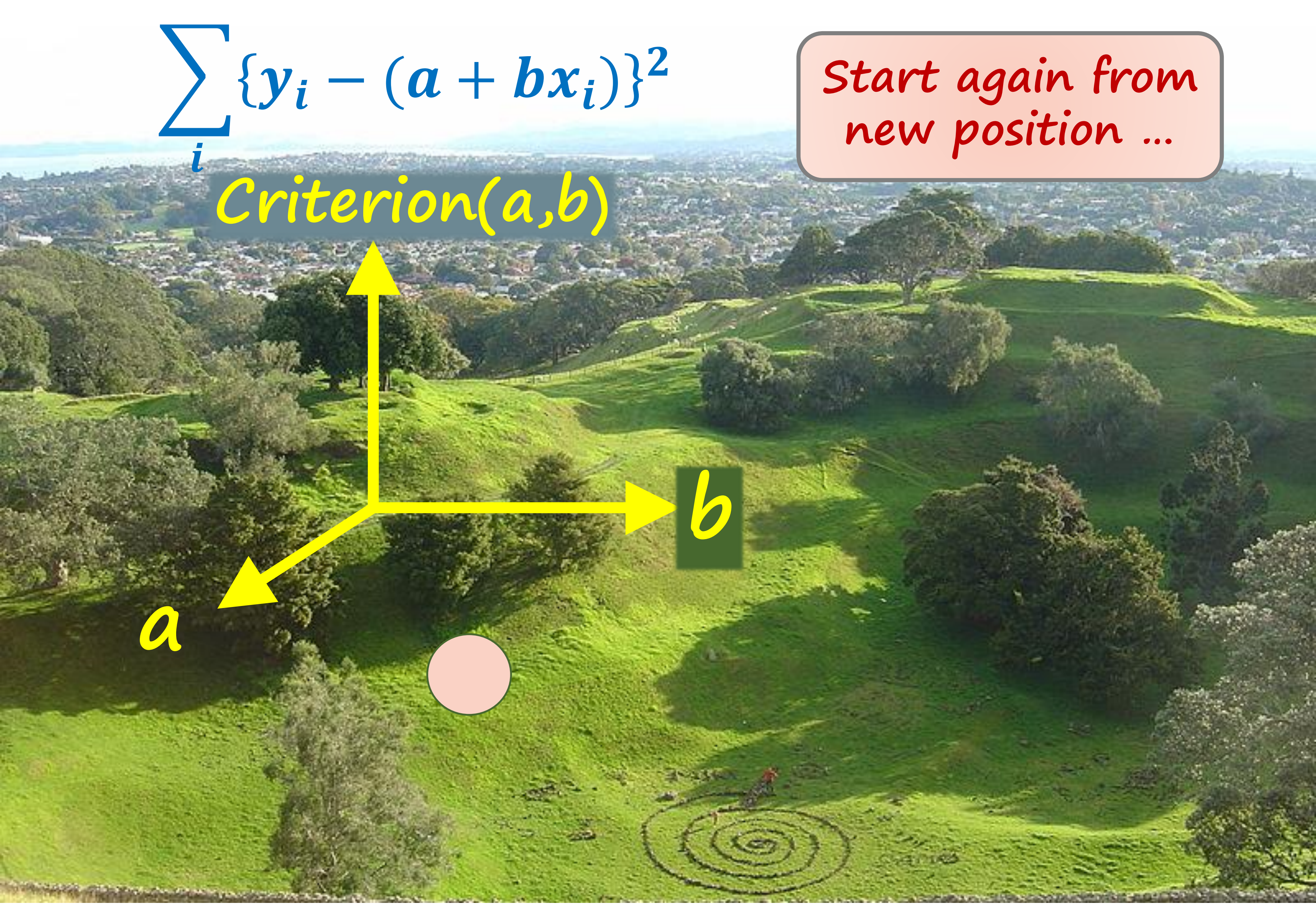
We're now a bit closer to the bottom …

… and all we did was calculate the criterion at different (a, b) !

We want to fumble our way downhill to the bottom

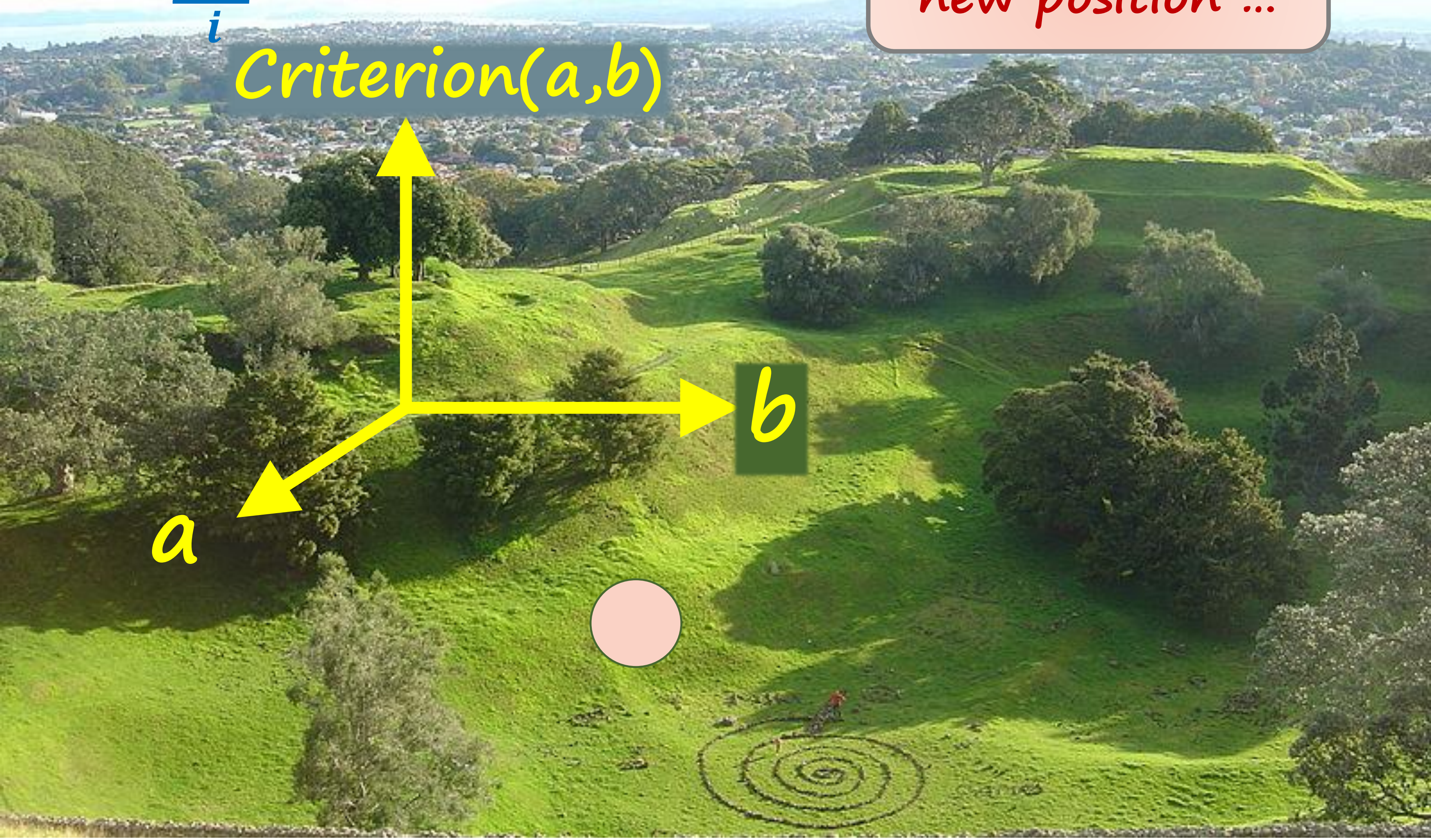


We want to fumble our way downhill to the bottom

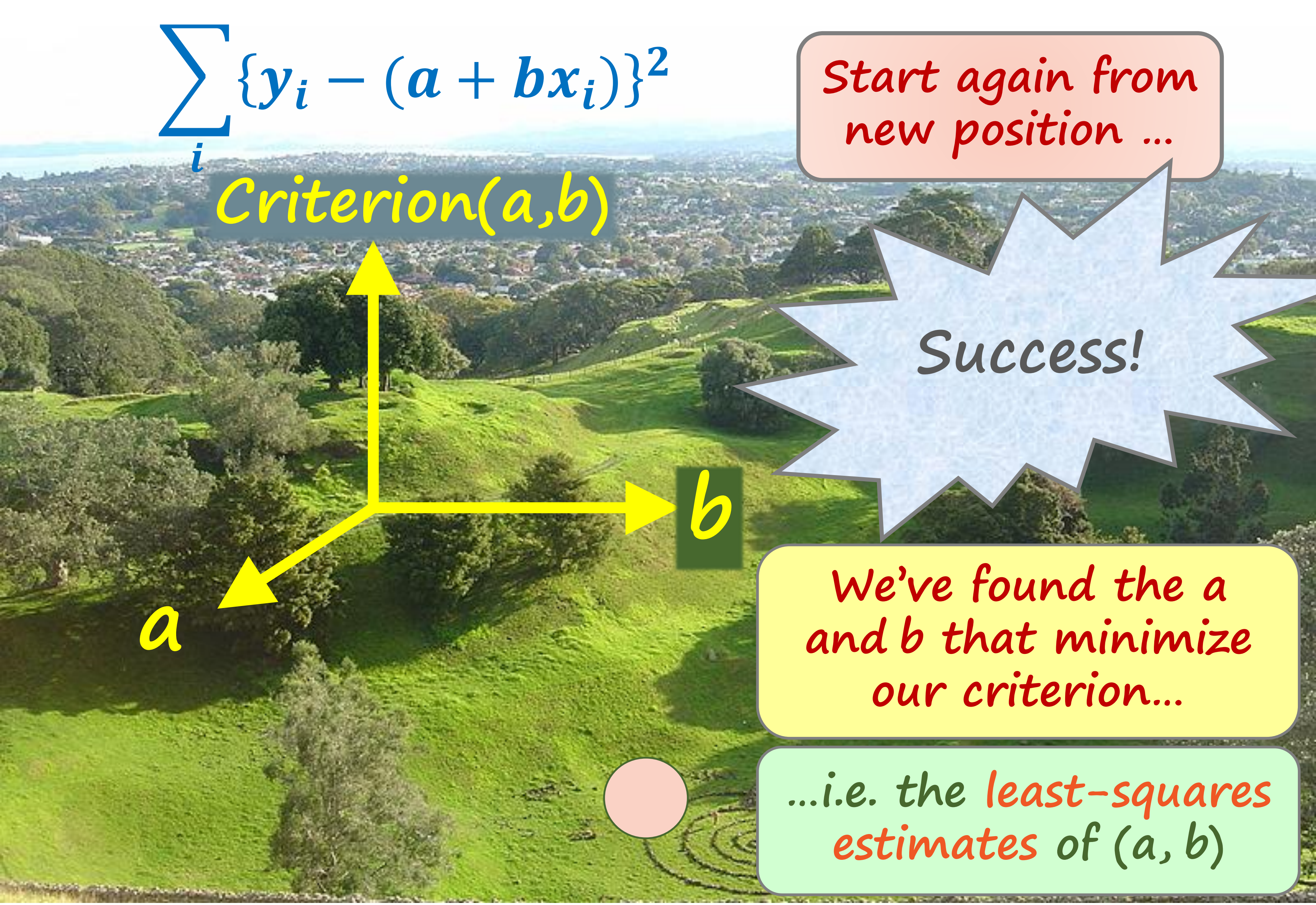


We want to fumble our way downhill to the bottom

$$\sum_i \{y_i - (a + bx_i)\}^2$$



We want to fumble our way downhill to the bottom



We want to fumble our way downhill to the bottom

How do we do this in R ?

Notice what we've achieved:

- **We can find the parameters (a, b) that optimize our criterion …**

- **… entirely by calculating the criterion at different choices of (a, b)**

  - **In R, if you can calculate it, you can optimize it**

To estimate parameters, we only need to be able to calculate some criterion that measures how good the fit is at each choice of parameters

How do we do this in R ?

- There are a few numerical optimizers in R:
  - nlm, nlminb, optim, …
- We recommend nlm as a good all-purpose optimizer to start with
  - nlm stands for non-linear minimization
- All it needs is:

1. A function to optimize: called the objective function or "criterion"
2. Somewhere to start: initial values of (a, b)

Minimize the criterion $f(a, b) = (a-1)^2 + (b-2)^2 + 3$:

```
findmin.func <- function(startvec=c(0, 0)){
        # Define the objective (criterion) as an inner function:
        objective.func <- function(pars){
                # The parameters have to be supplied as a vector:
                a <- pars[1]
                b <- pars[2]
                # Calculate the objective for this (a, b):
                return((a-1)^2 + (b-2)^2 + 3)
        }
         # Perform the minimization:
        nlm(f=objective.func, p=startvec)
}
```

Minimize the criterion $f(a, b) = (a-1)^2 + (b-2)^2 + 3$:

```
findmin.func <- function(startvec=c(0, 0)){
        objective.func <- function(pars){
                a <- pars[1]
                b <- pars[2]
                return((a-1)^2 + (b-2)^2 + 3)
        }
        nlm(f=objective.func, p=startvec)
}
```

Minimize the criterion $f(a, b) = (a-1)^2 + (b-2)^2 + 3$:

```
> findmin.func()
$minimum
[1] 3

$estimate
[1] 1 2

$gradient
[1] 0.000000e+00 1.110223e-12

$code
[1] 1

$iterations
[1] 2
```

Is this what you would expect?

Best values of a and b: the ones that minimize the objective

Check that the gradient is close to 0 at the minimum

What could possibly go wrong …?

```
findmin2.func <- function(startvec=c(0, 0)){
        objective.func <- function(pars){
                a <- pars[1]
                b <- pars[2]
                return(log(a-1)^2 + log(b-2)^2)
        }
        nlm(f=objective.func, p=startvec)
}
```

1. Bad start values!

```
> findmin2.func()
$minimum
[1] 1.797693e+308

$estimate
[1] 0 0

$gradient
[1] 0 0

$code
[1] 1

$iterations
[1] 0

There were 11 warnings (use warnings() to see them)
> warnings()
Warning messages:
1: In log(a - 1) : NaNs produced
2: In log(b - 2) : NaNs produced
3: In log(a - 1) : NaNs produced
4: In log(b - 2) : NaNs produced
5: In nlm(f = objective.func, p = startvec) :
  NA/Inf replaced by maximum positive value
```

1. Bad start values:

Solution is to choose better /alternative start values

```
> findmin2.func(startvec=c(5,5))
$minimum
[1] 3.430193e-12

$estimate
[1] 1.999999 2.999998

$gradient
[1]  2.436579e-08 -1.333134e-07

$code
[1] 1

$iterations
[1] 12
```

All looks good now

What could possibly go wrong …?

```
bad.func <- function(startvec=c(1, 1)){
        objective.func <- function(pars){
                a <- pars[1]
                b <- pars[2]
                return( (a-1)^2+ (b-2)^2 + 1/(a-b)^2 )
        }
        nlm(f=objective.func, p=startvec)
}
```

2. Some (a,b) combinations don't work: evaluate to NA or Inf

```r
fixed.func <- function(startvec=c(1, 1)){
	objective.func <- function(pars){
		a <- pars[1]
		b <- pars[2]
		obj <- (a-1)^2+ (b-2)^2 + 1/(a-b)^2
		# Print output so we can see what’s happening:
		print(c(a, b, obj))
		# Fix up the case where obj is NA or Inf by
		# redefining it as a very large positive number:
		if(is.na(obj) | is.infinite(obj))
			obj <- (abs(a)+abs(b))*1e10
		return(obj)
	}
	nlm(f=objective.func, p=startvec)
}
```

Your first task …

- Given a code template your first task is to complete the code and reproduce the output from the “lm” function
  - lm stands for linear model
  - lm is fitted in R by a least-squares criterion

- Coming next: how to use the bootstrap to compute a standard error for your least-squares estimates of (a, b) …

Standard error by bootstrap

We've said that the standard error is a type of quality assurance …

… but what does it really mean?

The standard error aims to measure the variability you'd see in estimates of (a, b) …

… if you conducted your whole estimation procedure again and again:

– starting with collecting your data

The standard error measures the variability you would see in your estimates of (a, b) …

… if you ran your whole estimation procedure over and over:

– starting with collecting your data

So in an ideal world we would have:

- Lots and lots of data sets;
- Each one has the same characteristics as our real data (same sample size and study design);
- We'd estimate (a, b) from each one…
- and just measure the variance in our estimates!

- Lots of data sets
- Each one has the same design as our real data
- Estimate (a, b) from each data set…
- Measure the variance in our estimates

This is exactly what the bootstrap aims to do

We only have ONE real data set …

… but if we resample from it, we could artificially create new data-sets that mimic our real data:

- Same sample size
- Same target population

Standard error by bootstrap

Original data

	X	Y
1	315.0	6.0
2	316.9	2.0
3	317.6	7.8
4	318.4	8.9
5	319.0	10.6
6	319.6	-14.9

Standard error by bootstrap

Original data

	X	Y
1	315.0	6.0
2	316.9	2.0
3	317.6	7.8
4	318.4	8.9
5	319.0	10.6
6	319.6	-14.9

	X	Y
6	319.6	-14.9
2	316.9	2.0
2	[illegible]	[illegible]
1	[illegible]	[illegible]
3	[illegible]	7.8
6	319.6	-14.9

Bootstrap replicate 1

	X	Y
5	319.0	10.6
5	319.0	10.6
1	[illegible]	[illegible]
4	[illegible]	[illegible]
5	[illegible]	10.6
2	316.9	2.0

Bootstrap replicate 2

	X	Y
3	317.6	7.8
1	315.0	6.0
2	[illegible]	[illegible]
1	[illegible]	[illegible]
3	[illegible]	7.8
2	316.9	2.0

Bootstrap replicate 3

B1

	X	Y
5	319.0	10.6
1	315.0	6.0
4	318.4	[illegible]
4	318.4	[illegible]
3	317.6	7.8
2	316.9	2.0

B2

	X	Y
3	317.6	7.8
5	319.0	10.6
1	315.0	[illegible]
4	318.4	[illegible]
5	319.0	10.6
1	315.0	6.0

B3

	X	Y
3	317.6	7.8
4	318.4	8.9
4	318.4	[illegible]
1	315.0	[illegible]
3	317.6	7.8
6	319.6	-14.9

B4

	X	Y
5	319.0	10.6
2	316.9	2.0
2	316.9	[illegible]
6	319.6	[illegible]
3	317.6	7.8
6	319.6	-14.9

And on and on and on!

Each bootstrap data-set mimics our real data:

- **Same sample size**
- **Same target population**

	X	Y
5	319.0	10.6
1	315.0	6.0
4	318.4	9
4	318.4	.9
3	317.6	7.8
2	316.9	2.0

$a_1\ b_1$

	X	Y
3	317.6	7.8
5	319.0	10.6
1	315.0	.0
4	318.4	.9
5	319.0	10.6
1	315.0	6.0

$a_2\ b_2$

	X	Y
3	317.6	7.8
4	318.4	8.9
4	318.4	9
1	315.0	0
3	317.6	7.8
6	319.6	-14.9

$a_3\ b_3$

	X	Y
5	319.0	10.6
2	316.9	2.0
2	316.9	0
6	319.6	.9
3	317.6	7.8
6	319.6	-14.9

$a_4\ b_4$

	X	Y
2	316.9	2.0
4	318.4	8.9
5	319.0	6
1	315.0	.0
3	317.6	7.8
1	315.0	6.0

$a_5\ b_5$

	X	Y
6	319.6	-14.9
5	319.0	10.6
6	319.6	9
4	318.4	.9
6	319.6	-14.9
2	316.9	2.0

$a_6\ b_6$

	X	Y
6	319.6	-14.9
2	316.9	2.0
2	316	2.0
1	315.	6.0
3	317.6	7.8
6	319.6	-14.9

$a_7\ b_7$

	X	Y
5	319.0	10.6
5	319.0	10.6
1	315.	.0
4	318.4	8.9
5	319.0	10.6
2	316.9	2.0

$a_8\ b_8$

Each one gives us an estimate of (a, b) ...

Our bootstrap estimates of the standard errors of a and b are:

$$\text{se}(a) = \text{sd}(a_1, a_2, \ldots\ldots, a_B)$$

$$\text{se}(b) = \text{sd}(b_1, b_2, \ldots\ldots, b_B)$$

2	316.9	2.0
6	319.6	-14.9
3	317.6	7.8
6	319.6	-14.9

5	319.0	10.6
1	315.0	6.0
3	317.6	7.8
1	315.0	6.0

		.6
6	319.6	-14.9
4	318.4	8.9
6	319.6	-14.9
2	316.9	2.0

	X	Y
6	319.6	-14.9
2	316.9	2.0
2	316	2.0
1	315.	6.0
3	317.6	7.8
6	319.6	-14.9

$a_7 \; b_7$

	X	Y
5	319.0	10.6
5	319.0	10.6
1	315.	.0
4	318.4	8.9
5	319.0	10.6
2	316.9	2.0

$a_8 \; b_8$

Each one gives us an estimate of (a, b) …

Equivalently:

se(a) = sqrt(var($boot\$a$))

se(b) = sqrt(var($boot\$b$))

2	316.9	2.0
6	319.6	-14.9
3	317.6	7.8
6	319.6	-14.9

	X	Y
6	319.6	-14.9
2	316.9	2.0
2	316	2.0
1	315.	6.0
3	317.6	7.8
6	319.6	-14.9

a_7 b_7

5	319.0	10.6
1	315.0	6.0
3	317.6	7.8
1	315.0	6.0

	X	Y
5	319.0	10.6
5	319.0	10.6
1	315.	.0
4	318.4	8.9
5	319.0	10.6
2	316.9	2.0

a_8 b_8

		.6
6	319.6	-14.9
4	318.4	8.9
6	319.6	-14.9
2	316.9	2.0

Each one gives us an estimate of (a, b) ...

For 95% confidence intervals:

quantile($boot\$a$, $probs$=c(0.025,0.975))

quantile($boot\$b$, $probs$=c(0.025, 0.975))

2	316.9	2.0
6	319.6	-14.9
3	317.6	7.8
6	319.6	-14.9

5	319.0	10.6
1	315.0	6.0
3	317.6	7.8
1	315.0	6.0

6	319.6	-14.9
4	318.4	8.9
6	319.6	-14.9
2	316.9	2.0

	X	Y
6	319.6	-14.9
2	316.9	2.0
2	316	2.0
1	315.	6.0
3	317.6	7.8
6	319.6	-14.9

a_7 b_7

	X	Y
5	319.0	10.6
5	319.0	10.6
1	315.	.0
4	318.4	8.9
5	319.0	10.6
2	316.9	2.0

a_8 b_8

Each one gives us an estimate of (a, b) ...

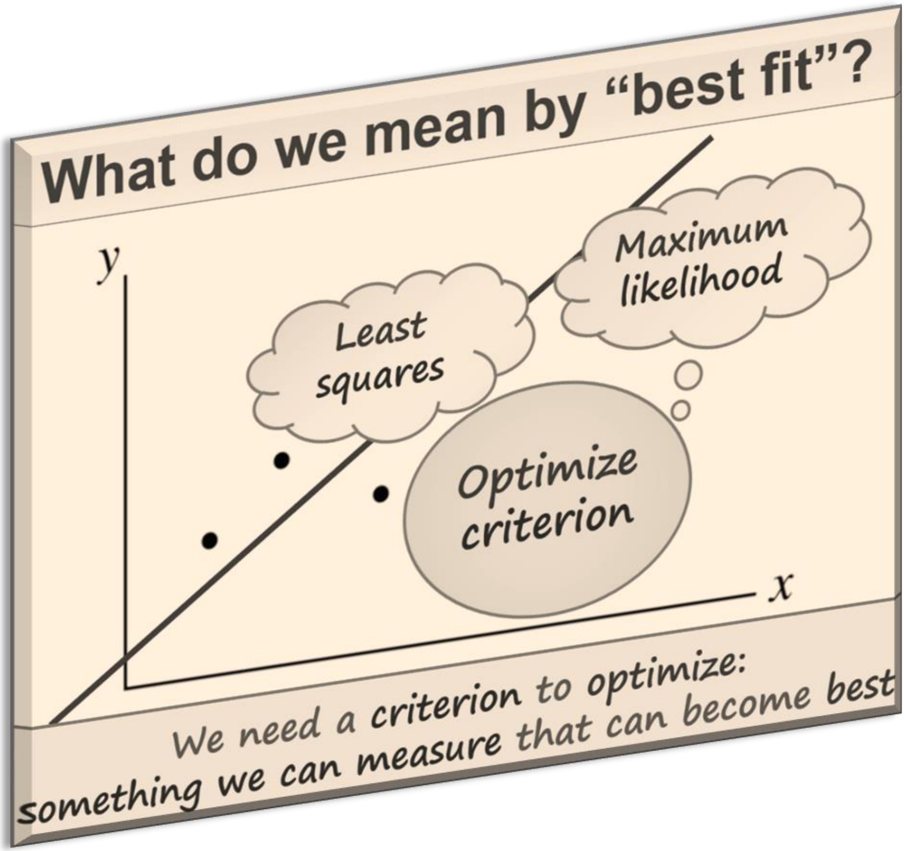
```
bootstrap.func <- function(dat, B=1000, startvec=c(0, 0)){
  # Create empty boot.res for storing your results:
  boot.res <- data.frame(rep=1:B, a=rep(NA, B), b=rep(NA, B))
  # Loop for bootstrap replicates:
  for(i in 1:B){
        # Sample the rows at random with replacement:
        resampleRows <- sample(1:nrow, size=nrow, replace=T)
        # Create the data for the resample:
        dat.boot <- dat[resampleRows, ]
        # Fit the model to this replicate:
        fit.boot <- leastSquares.func(dat.boot, startvec)
        # Enter the estimated values into row i of boot.res:
        boot.res$a[i] <- fit.boot$estimate[1]
        boot.res$b[i] <- fit.boot$estimate[2]
   } # End of loop
  # Find the confidence intervals:
  CI.a <- quantile(boot.res$a, probs=c(0.025, 0.975))
  CI.b <- quantile(boot.res$b, probs=c(0.025, 0.975))
}
```

```r
bootstrap.func <- function(dat, B=1000, startvec=c(0, 0)){
  n <- nrow(dat)
  boot.res <- data.frame(rep=1:B, a=rep(NA, B), b=rep(NA, B))
  for(i in 1:B){
        resampleRows <- sample(1:n, size=n, replace=T)
        dat.boot <- dat[resampleRows, ]
        fit.boot <- leastSquares.func(dat.boot, startvec)
        boot.res$a[i] <- fit.boot$estimate[1]
        boot.res$b[i] <- fit.boot$estimate[2]
   }
  # Find the confidence intervals:
  CI.a <- quantile(boot.res$a, probs=c(0.025, 0.975))
  CI.b <- quantile(boot.res$b, probs=c(0.025, 0.975))
  return(list(a=CI.a, b=CI.b))
}
```

DIY Modelling – we're nearly there!

- We can fit the model: i.e. estimate a and b

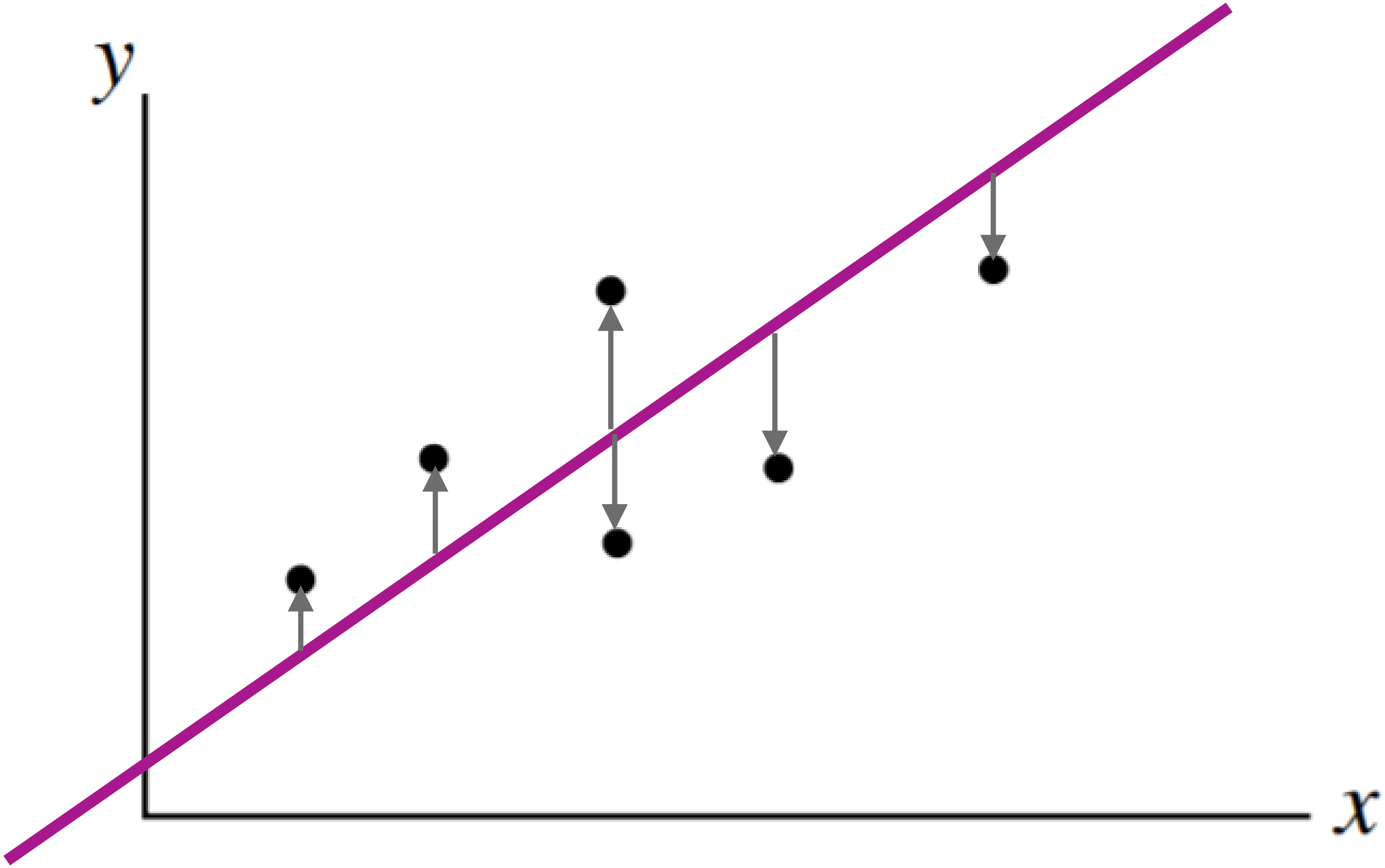
- We can compute their standard errors



One more thing:

Can we improve on the Least Squares criterion?

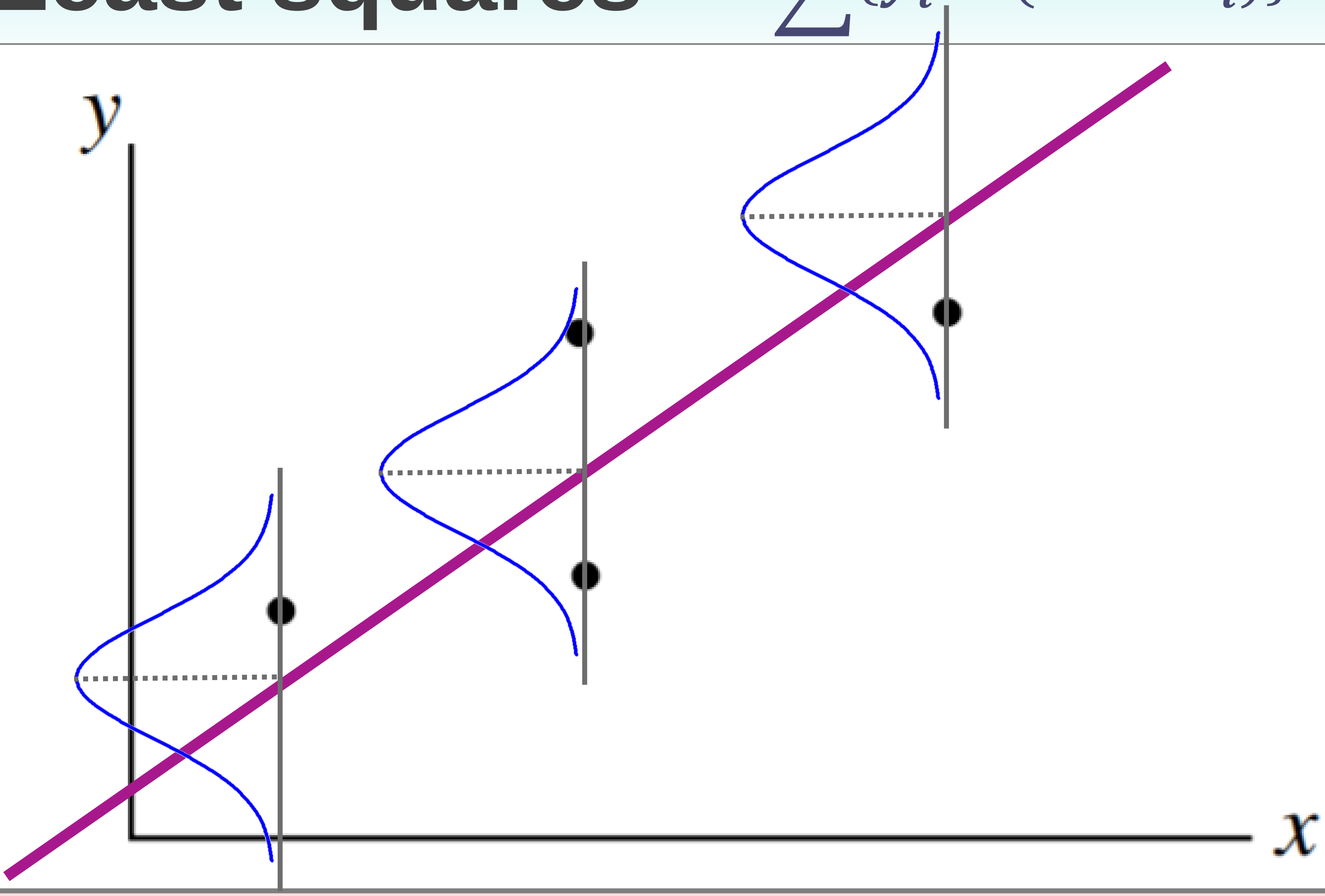
Least squares $\sum\{y_i - (a + bx_i)\}^2$



Least squares assumes the data are scattered symmetrically and evenly about the line

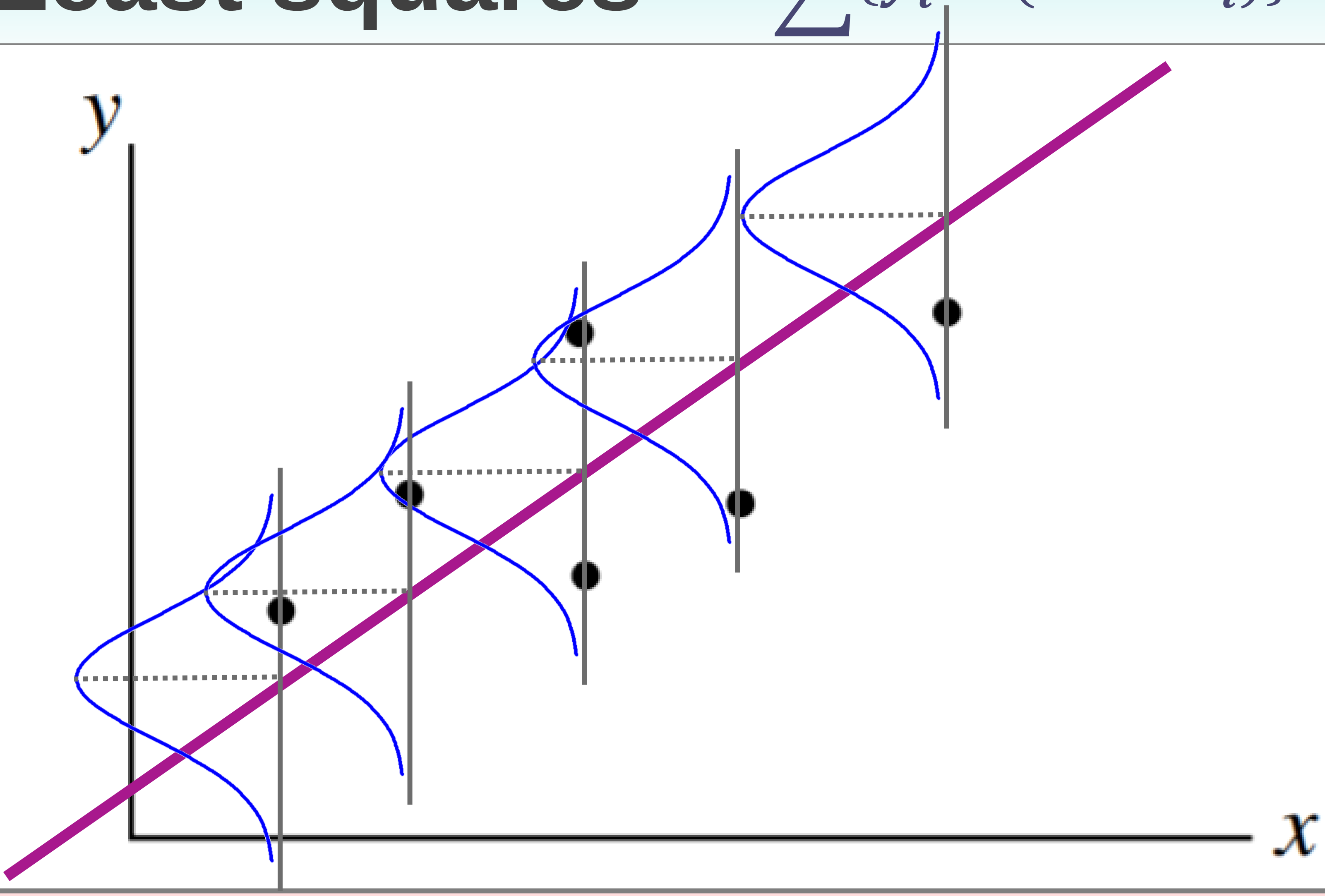
Least squares

$$\sum \{y_i - (a + bx_i)\}^2$$



Least squares assumes the data are scattered symmetrically and evenly about the line

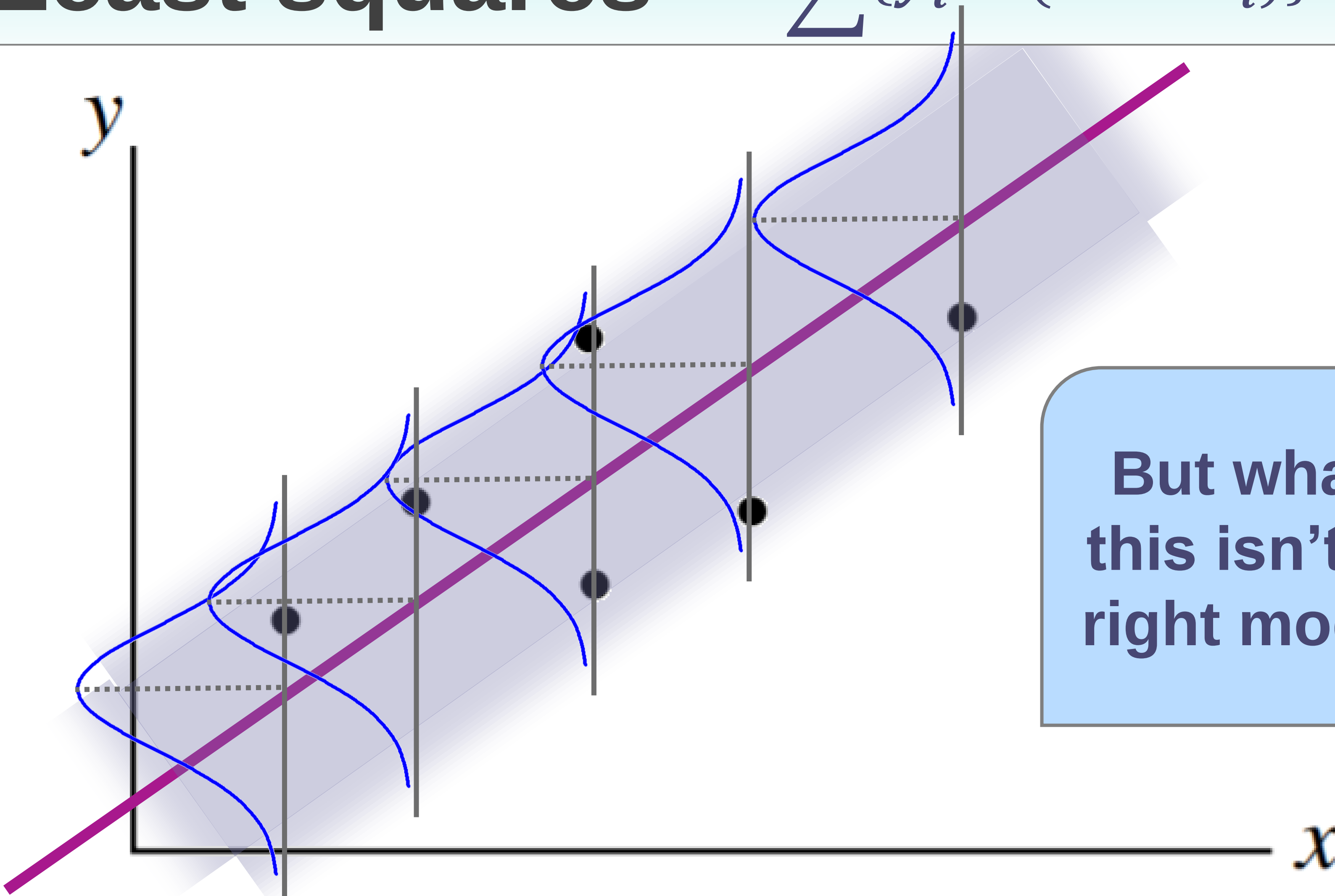
Least squares $\sum \{y_i - (a + bx_i)\}^2$



Least squares assumes the data are scattered symmetrically and evenly about the line

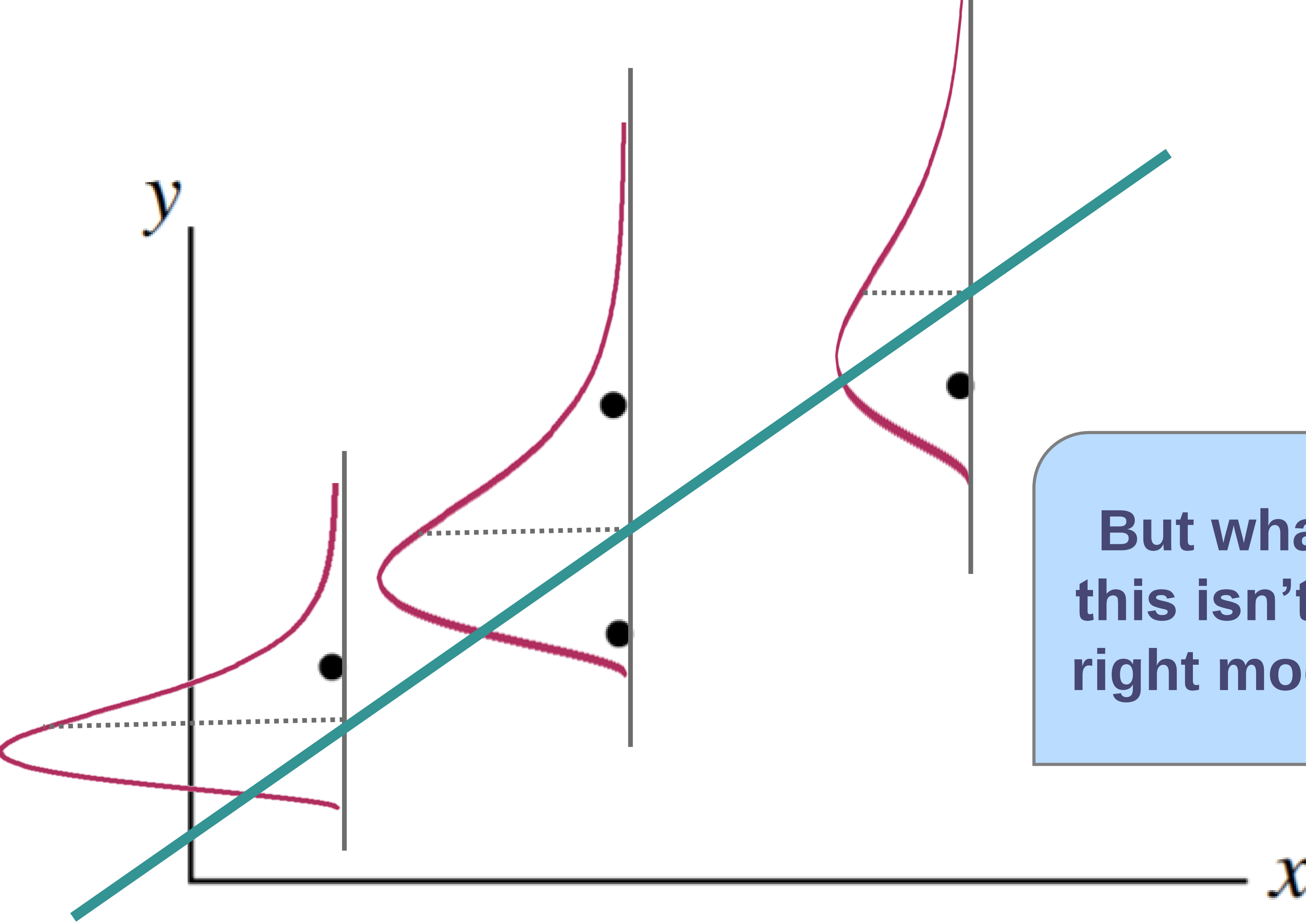
Least squares

$$\sum\{y_i-(a+bx_i)\}^2$$



But what if this isn't the right model?

Least squares assumes the data are scattered symmetrically and evenly about the line

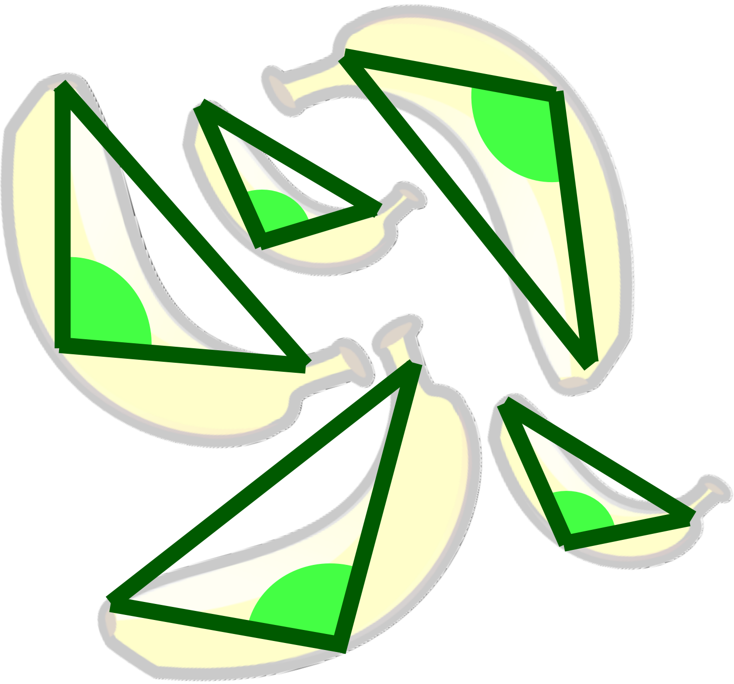


But what if this isn’t the right model?

Often a better model would allow right skew and/or scatter that increases along the line

Or maybe there isn’t even a line...

How bent is a banana?

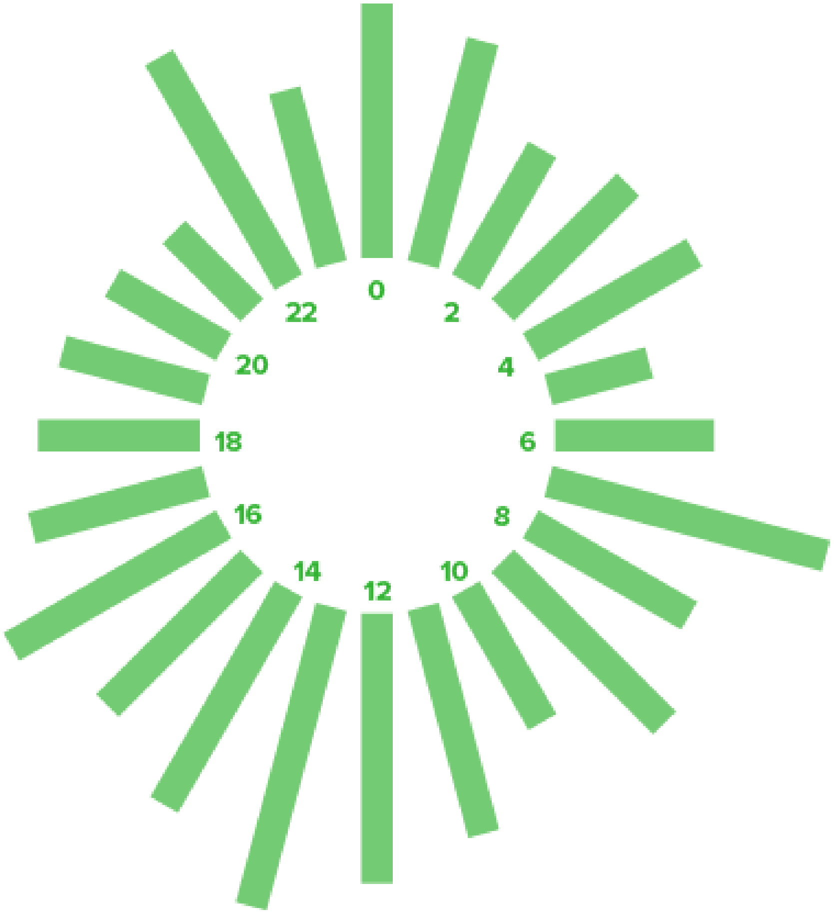


How many kiwi are there in the Coromandel?

We often want to fit models that don’t involve scatter about a line at all

Or maybe there isn't even a line...

What time of day do I receive emails?



We often want to fit models that don't involve scatter about a line at all

Maximum likelihood estimation

All these models have one thing in common:

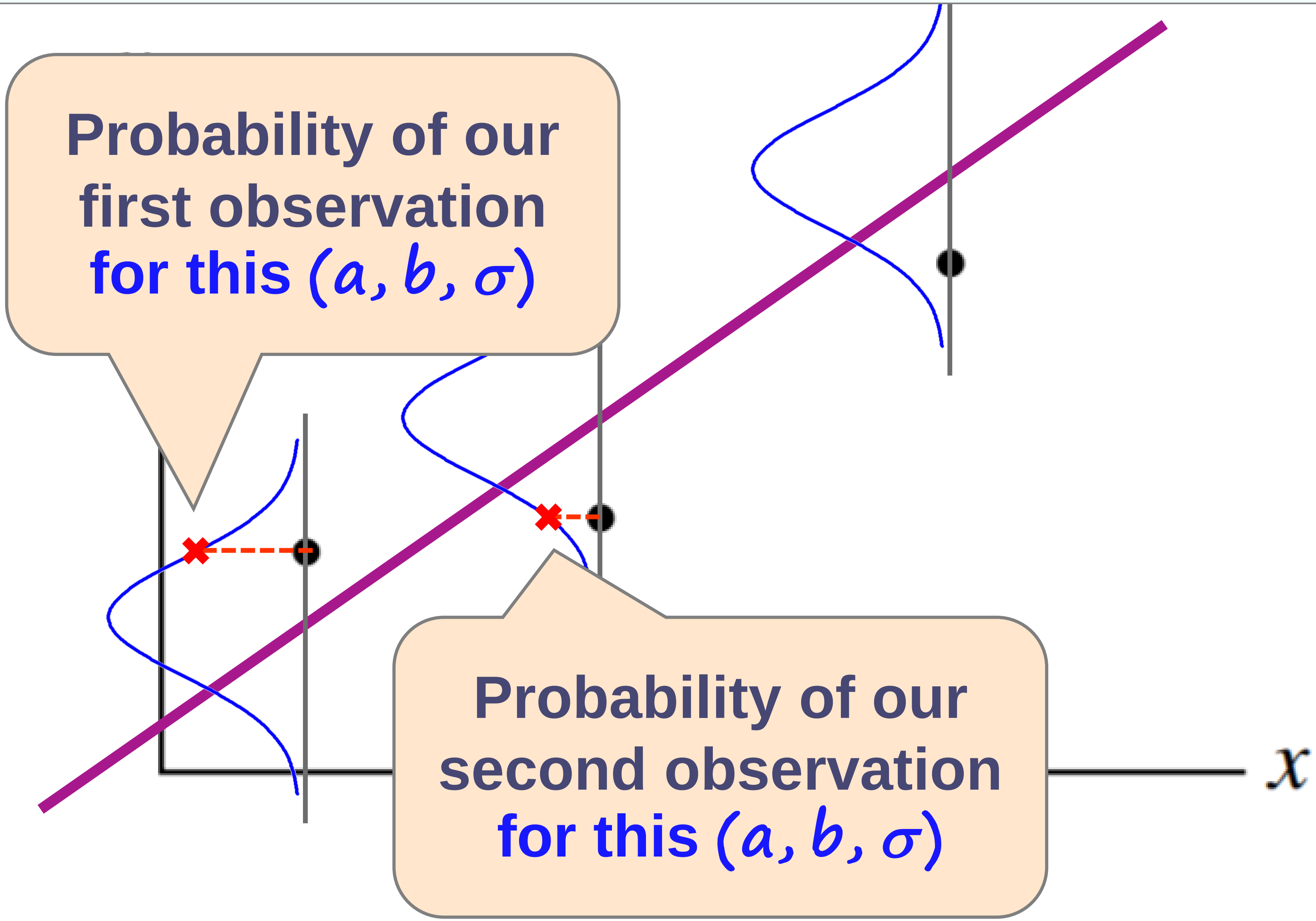
… a statistical distribution or stochastic process that we imagine our data are generated from

… under which we can calculate the probability of our data…

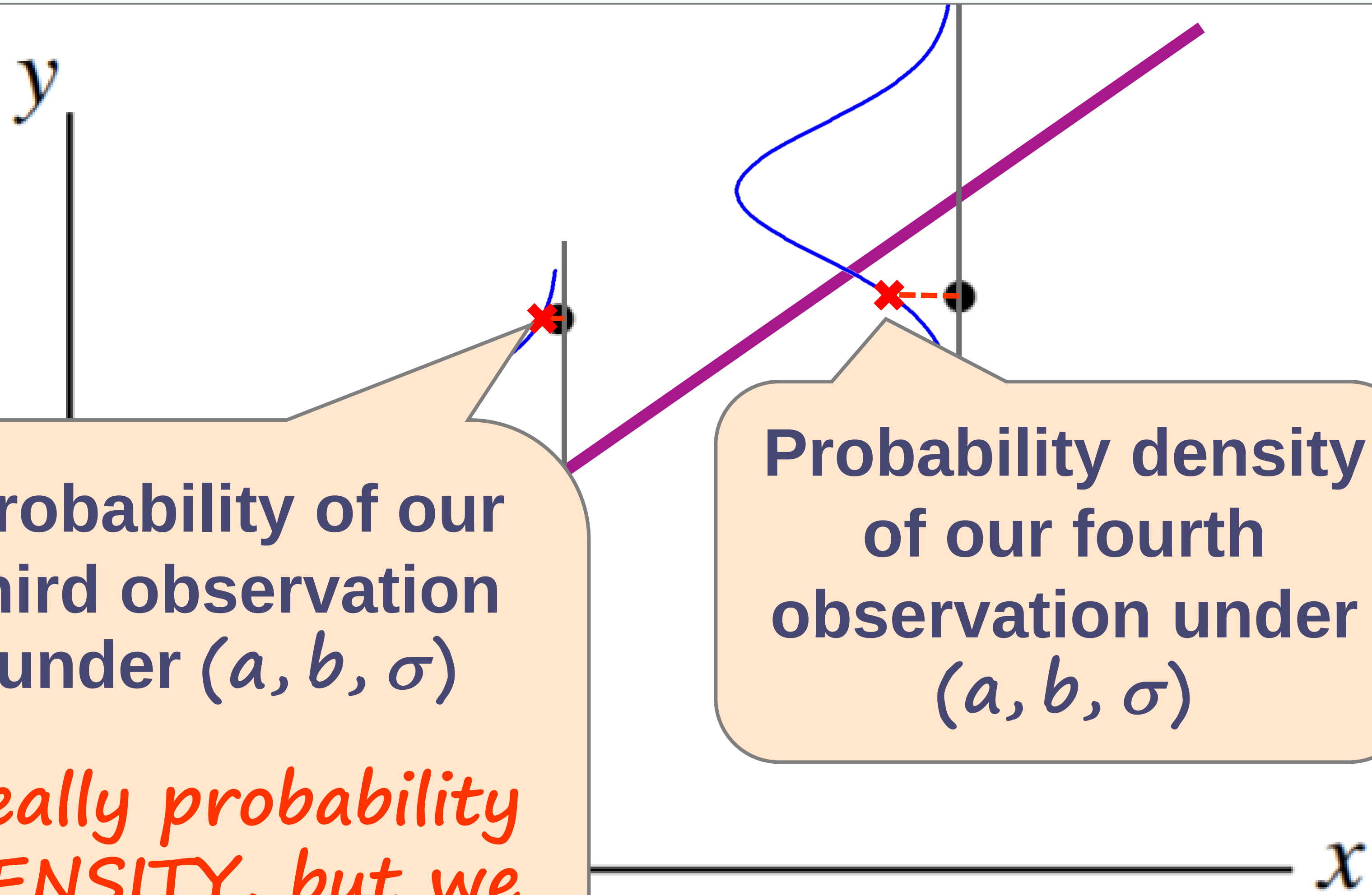
Why not use this probability as our fit criterion?

- **Good parameter values are those which give our data high probability**
- **a and b are good estimates if they make our data highly likely**

Maximum likelihood estimation



Maximum likelihood estimation

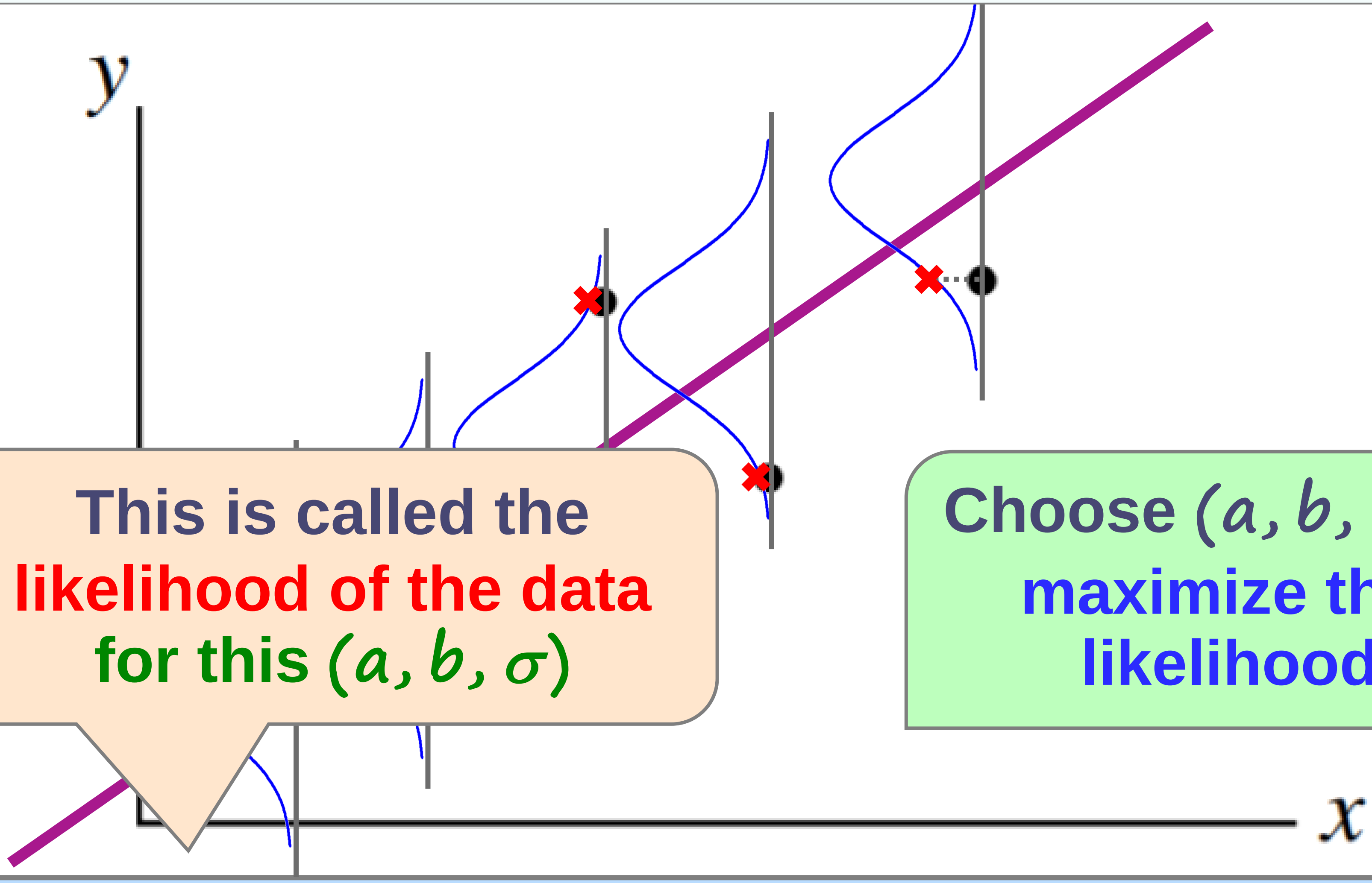


Probability of our third observation under (a, b, σ)

Really probability DENSITY, but we can think of it like a probability

Probability density of our fourth observation under (a, b, σ)

Maximum likelihood estimation



This is called the **likelihood of the data** for this (a, b, σ)

Choose (a, b, σ) to **maximize this likelihood**

Multiply together to get the overall probability density of the observations for this (a, b, σ)

Maximum likelihood estimation

In practice we don't multiply the probabilities:

- **Multiplying tens or hundreds of small numbers will create computational problems**

- **Instead, take logs and add:**

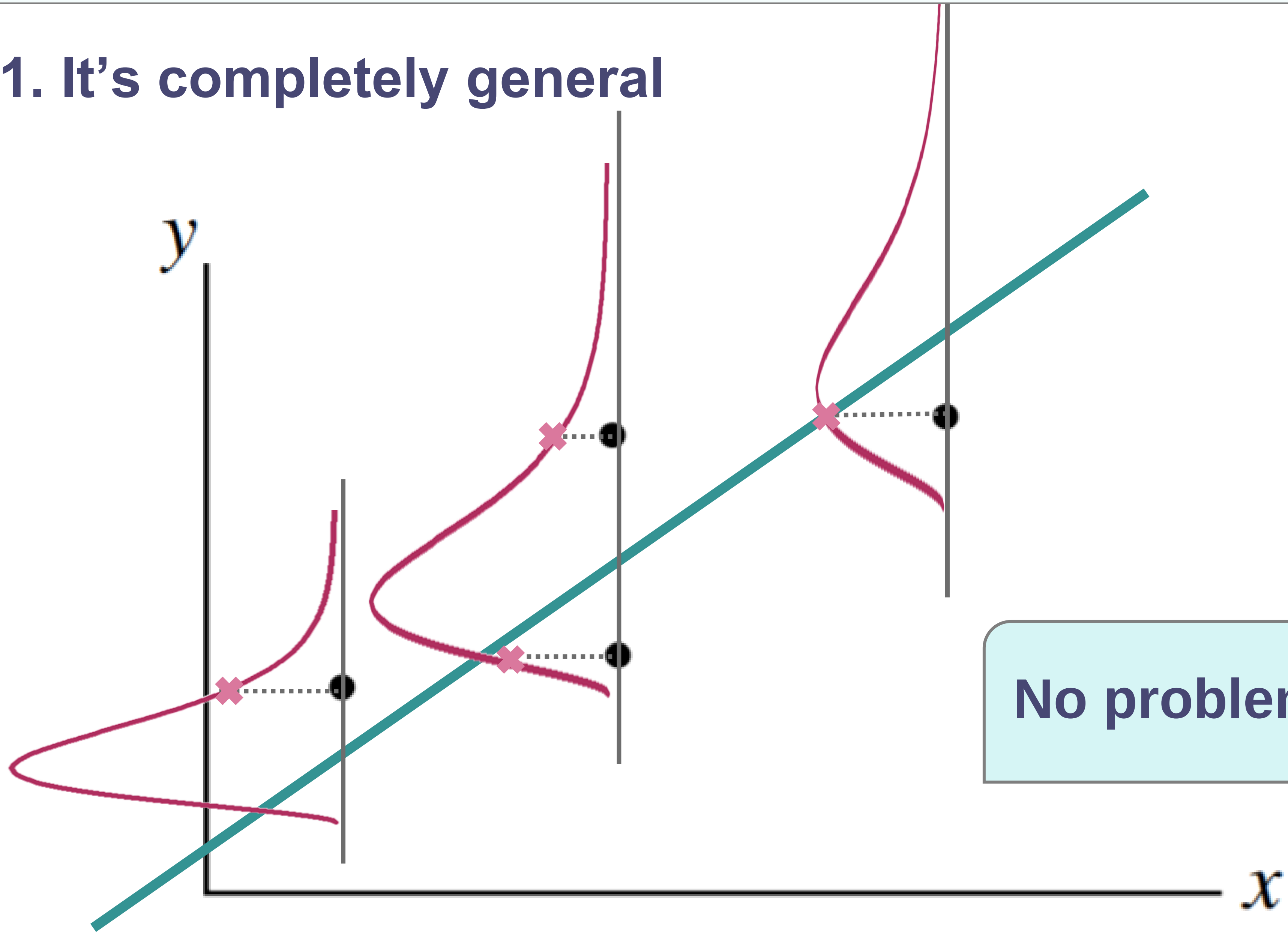
$$\log\left\{\prod_i P(y_i)\right\} = \sum_i \log\{P(y_i)\}$$

- **The log-likelihood is maximized at the same (a, b, σ) as the likelihood**

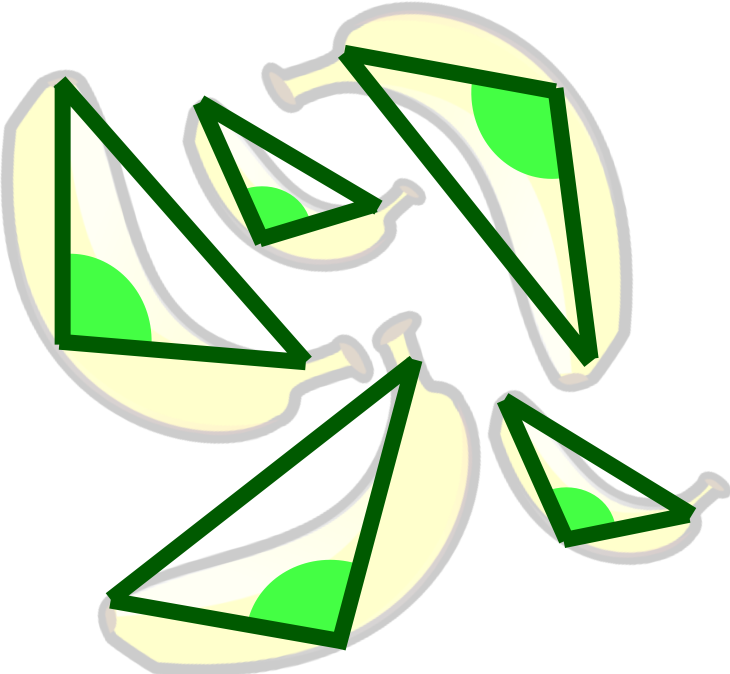
- **Can be easily computed**

Why use maximum likelihood?

1. It's completely general



No problem!



Maximum likelihood works for everything!

Why use maximum likelihood?

2. It's usually the best possible choice:

- **Maximum likelihood estimation yields the lowest possible standard error**

The standard error measures the variability you would see in your estimates of (a, b) …

… if you conducted your whole estimation procedure over and over

Some estimation procedures are more efficient than others!

Maximum likelihood in R

- **You can only minimize functions in R, not maximize them**

- **So minimize the negative log likelihood**

- **Least squares objective is:**

sum((yData – yPredicted)^2)

- **Maximum likelihood objective is:**

– sum(dnorm(yData, mean=yPredicted, sigma, log=T))

Negative log-likelihood is a sum …

…. of log-likelihoods for each observation

Maximum likelihood in R

- **You can only minimize functions in R, not maximize them**

- **So minimize the negative log likelihood**

- **Least squares objective is:**

  sum((yData – yPredicted)^2)

- **Maximum likelihood objective is:**

– sum(dnorm(yData, mean=yPredicted, sigma, log=T))

“dnorm” means use the probability density from the Normal distribution

Maximum likelihood in R

- **You can only minimize functions in R, not maximize them**

- **So minimize the negative log likelihood**

- **Least squares objective is:**

  sum((yData – yPredicted)^2)

- **Maximum likelihood objective is:**

– sum(dnorm(yData, mean=yPredicted, sigma, log=T))

- **For the Normal distribution, maximum likelihood gives the same answers as least squares!**

Want to change your model?

- **Easy!**

- **Normal model objective is:**

– sum(dnorm(yData, mean=yPredicted, sigma, log=T))

- **Poisson model objective is:**
  - – sum(dpois(yData, rate=yPredicted, log=T))

- **You can calculate the standard error by bootstrap as before**

DIY Modelling: Your tasks

- The code template contains incomplete R code to fit an “lm” model by least squares and find the standard error by bootstrap

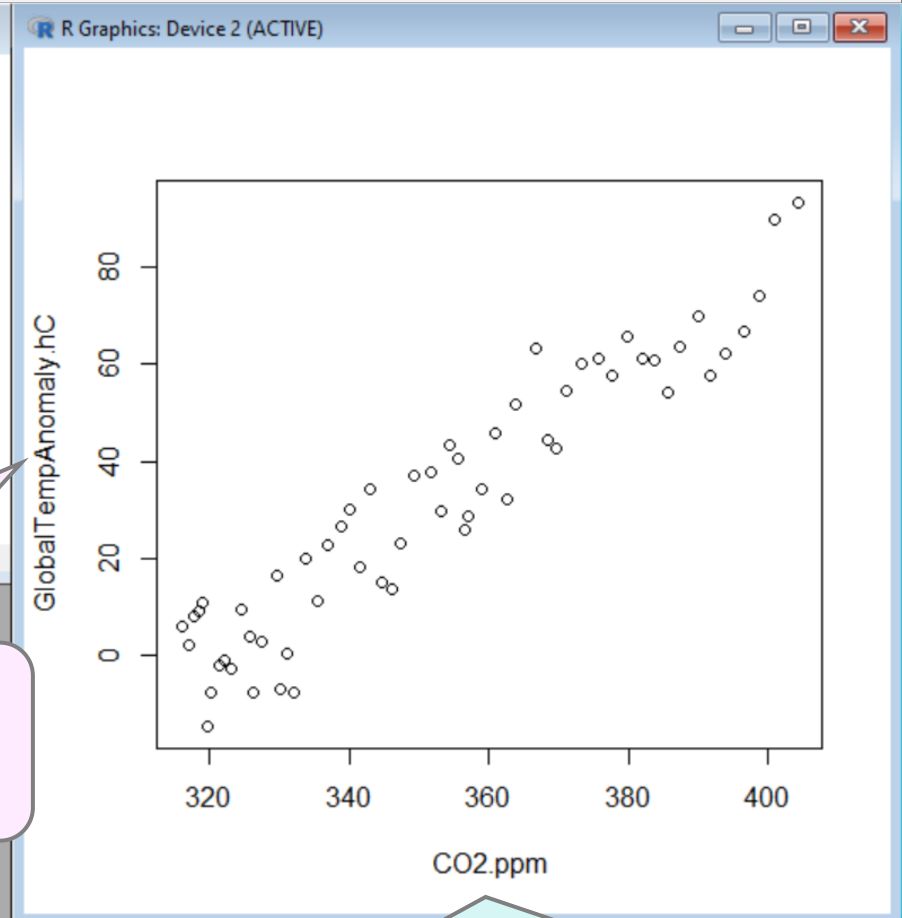
- First job: complete the code
  - Apply it to the Climate Data provided
  - Demonstrate you get the same estimates as lm, and similar standard errors / CIs
  - Write new code to fit the same model by maximum likelihood (Normal scatter model)
  - Demonstrate you get the same answers again

1. Climate data 1959-2016

GlobalTemperatureAnomaly.csv : call it temp.dat

R Console

```
> temp.dat <- read.csv(file.choose())
> head(temp.dat)
  CO2.ppm GlobalTempAnomaly.hC
1  315.97                 5.96
2  316.91                 2.04
3  317.64                 7.75
4  318.45                 8.88
5  318.99                10.68
6  319.62               -14.95
>
> with(temp.dat, plot(CO2.ppm, GlobalTempAnomaly.hC))
>
```



Global temperature anomaly in hC (hundredths of ºC)

Looks like good data for an “lm” model

Annual mean atmospheric CO_2 in ppm (parts per million)

2. Wombat data

Wombats.csv : call it wombat.dat

- The northern hairy-nosed wombat lives in just two locations in Queensland
- One of the world's rarest mammals: about 250 total
- Every few years there is a burrow survey to estimate the population size

- Sticky tapes erected outside burrows catch wombat hairs as the wombats go out for the night

2. Wombat data

Wombats.csv : call it wombat.dat

Photos: Dr Alan Horsup

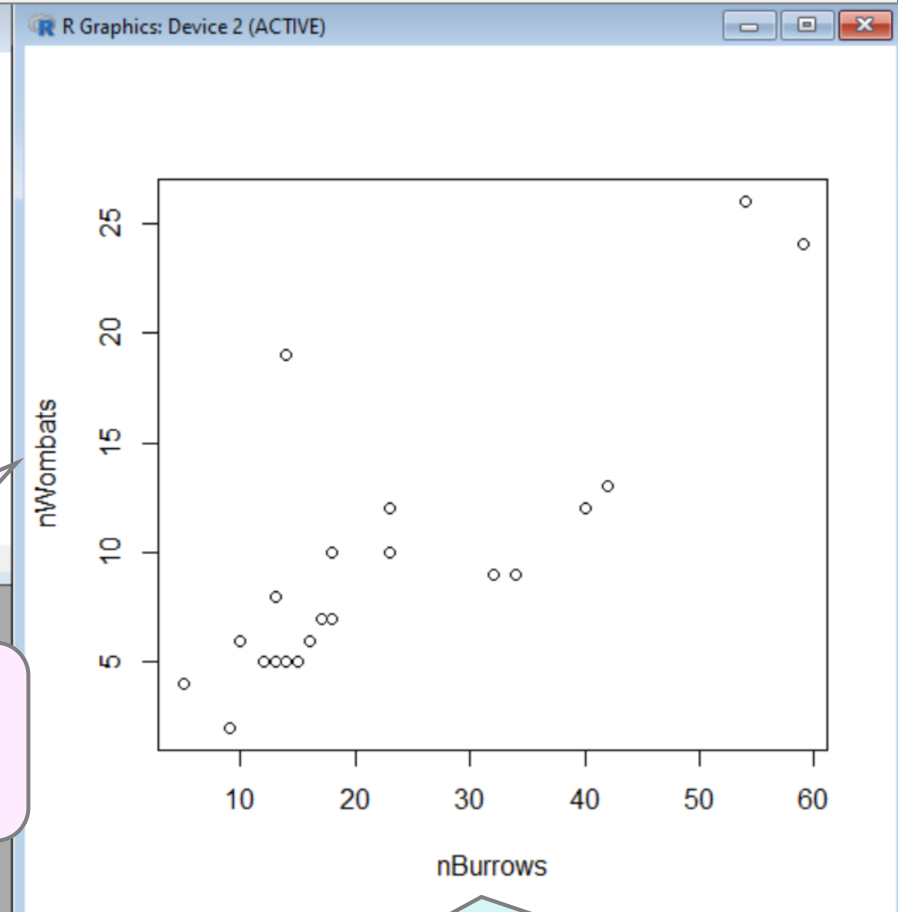
- **Sticky tapes erected outside burrows catch wombat hairs as the wombats go out for the night**

2. Wombat data

Wombats.csv : call it wombat.dat

R Console

```
>
> wombat.dat <- read.csv(file.choose())
> head(wombat.dat)
  location nBurrows nWombats males females
1     Bend       32        9     3       6
2     Camp       59       24    12      12
3     Cave       34        9     7       2
4  Coombes       23       10     4       6
5    Davis       15        5     2       3
6   Dexter       18        7     5       2
> with(wombat.dat, plot(nBurrows, nWombats))
> |
```



Number of different wombats sampled from that location

Looks like good data for a Poisson glm

#burrows in each location (location = cluster of burrows)

2. Wombat data

Wombats.csv : call it wombat.dat

```
> wombat.dat <- read.csv(file.choose())
> head(wombat.dat)
  location nBurrows nWombats males females
1     Bend       32        9     3       6
2     Camp       59       24    12      12
3     Cave       34        9     7       2
4  Coombes       23       10     4       6
5    Davis       15        5     2       3
6   Dexter       18        7     5       2
```

Maybe these columns could be suitable for some other type of model…?

DIY Modelling: your second task

- **Write MLE code to reproduce the following GLM using the Wombat data:**

glm(nWombats ~ nBurrows, family=poisson(link=log))

  - **Fit the GLM with poisson and quasipoisson:** glm(…, family=poisson) and glm(…, family=quasipoisson)
  - **Demonstrate your MLE code gives the same estimates as both poisson and quasipoisson**
  - **Investigate how the standard errors compare: are the bootstrap standard errors similar to the poisson or the quasipoisson ones? Why?**
  - **Write up all your findings in your report and submit your code on Canvas**

DIY Modelling: your third task

- **Do something else!**
- **Your ‘something else’ should use your DIY modelling skills in some way**
  - Code your own models

Ideas for your Something Else:

- Formulate a model for the extra columns in wombat.dat: code it and compare with R
- Find a context (data and/or model) where maximum likelihood is demonstrably better than least-squares
- Refresh theory from Stats 310 about how to calculate the standard error analytically: code it and compare it with your bootstrap results
- Code up the parametric bootstrap and compare with nonparametric bootstrap for both of the data-sets supplied

Report and video

- **Submit a brief project report (4 pages max)**
- **Compare output from R (lm / glm) with output from your own code (least-squares / MLE)**
  - Clearly show how the outputs demonstrate that your code matches the results from the R functions
  - Make sure you've answered all specific questions with each data-set (e.g. poisson *vs* quasipoisson standard errors)
- **Describe your Something Else and show output** (include points of interest, e.g. if a slope is significant)
- **On your video, show your code running in real time** (just the fitting code; the bootstrap may take too long)

Assessment: 8% total

- **Code that successfully answers the questions set for Tasks 1 and 2: 4% instructor**
  - We must be able to run your code successfully
- **Your Something Else: 4%**
  - 50% peer, 50% instructor

Good luck! ☺