How to avoid the ~~7~~ 5 deadly sins of data analysis.

Steven C. Wofsy

PKU-Harvard Summer school

Beijing, China

03 August 2017

Some of the most common errors made by scientists and engineers analyzing and modeling atmospheric and environmental data?

1. Forget/ignore the basic hypotheses underlying statistical inference from data.
2. Incorrectly fit a ”line” (model) to data (1): neglect the symmetry between “predictors” and “response” when assessing uncertainties the analysis
3. Incorrectly fit a ”line” (model) to data (2): fail to account for autocorrelation of data or predictors
4. Conflate standard deviation, uncertainty, and confidence intervals in an analysis
5. Not remember why the best way to assess uncertainties is almost always the bootstrap, or its big cousin, the Markov Chain Monte Carlo simulation

1. Statistical inference

The process we want to understand can be represented by a quantitative model.

Our measurements represent *samples* of the results of this process, modified by "errors" (usually with zero mean = "unbiased") which can be represented as a probability distribution

We can never know the exact values of the parameters of this model, or the exact description of the probability distribution function for the "errors".
Statistical inference provides *estimates* of the parameters and the pdf.

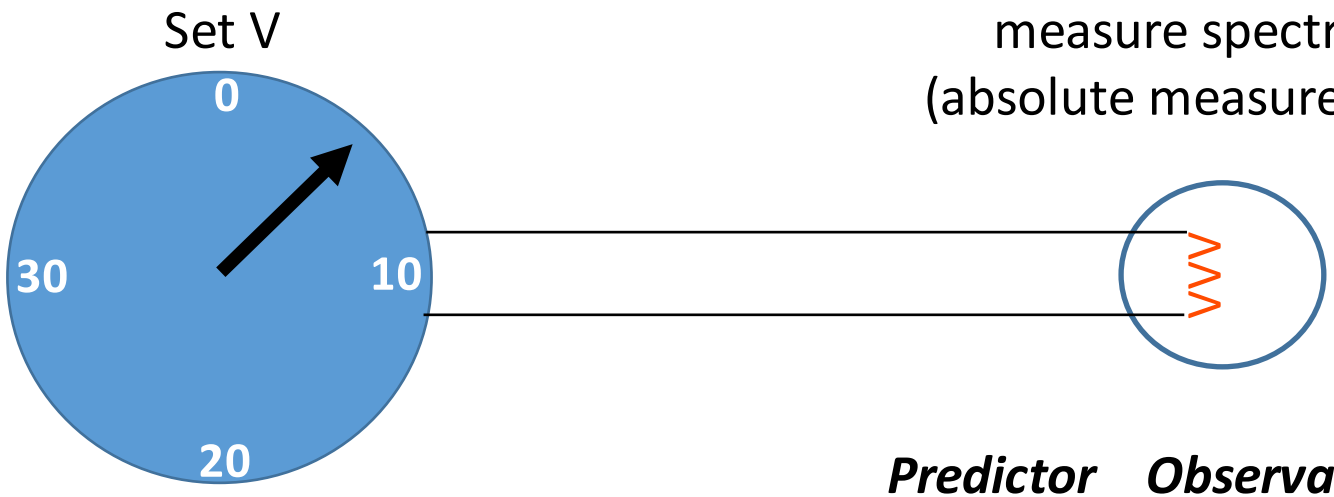
A common approach is to obtain an estimate of the *likelihood* of a set of model parameters and pdf given a set of measured values, and then to select as the "best estimate" the parameters and pdf that give the *maximum likelihood*.

1. **The framework is hypothetical: there exists a perfect model and a perfect probability distribution.**
2. **Our best model is an *estimate*, as is our set of "uncertainties".**

➔ It is possible to get wrong/biased estimates if my model of the process, or of the errors, is not correct for the system at hand.

2. Model, predictors, response; data—Symmetry?

Find resistance for the filament of a lightbulb, as a function of T:
turn a knob to set the voltage, observe filament T with a spectrometer



Predictor	***Observation***	***Model_Param***
$5 \pm \delta$ v	$1000 \pm \varepsilon$ K	$100 \pm \varepsilon'$ Ω
7	1155	110
…	…	…
		$a\sigma = 2.5 \times 10^{-13} \pm \varepsilon''$

$$E = V^2/R = a\sigma T^4$$

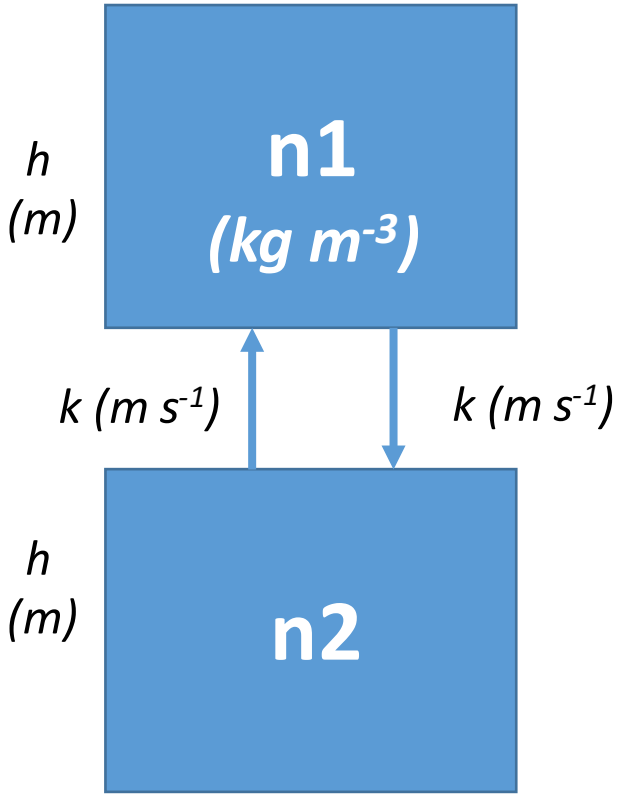
$$R = V^2/(a\sigma T^4)$$

response *predictors*

$\sigma = 5.670367 \times 10^{-8}$ kg s^{-3} K^{-4}

But the data are reversible (invertible)/symmetric: given observed E and resistance, infer voltage: "Inverse modeling" ⇔ "causality"

A working example: a 2-box model mass transfer model, and equivalent Markov chain



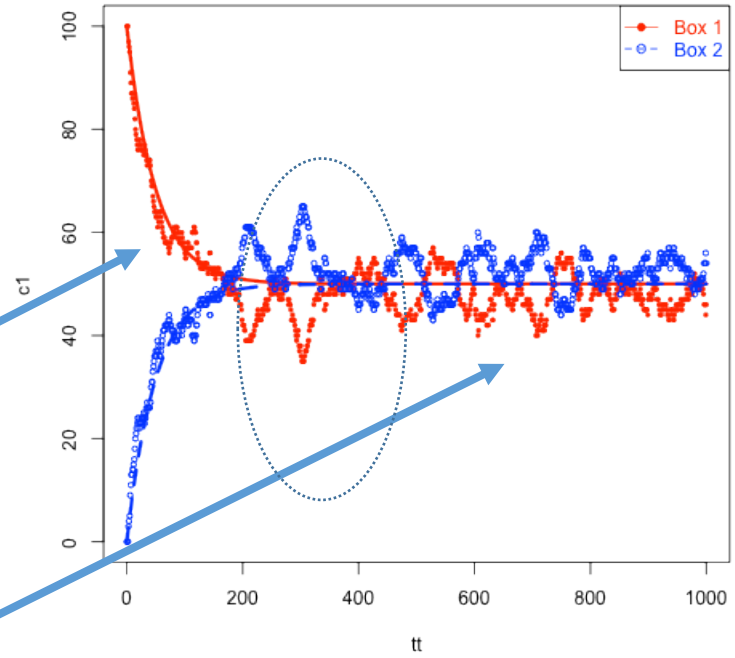
$mass_i$ = h Area n_i ; set k/h=.01

h dn1/dt = k (n2 –n1)
h dn2/dt = k (n1 –n2)

(n1 – n2) = d = d_o exp(-2k/h t)
n1 + n2 = constant

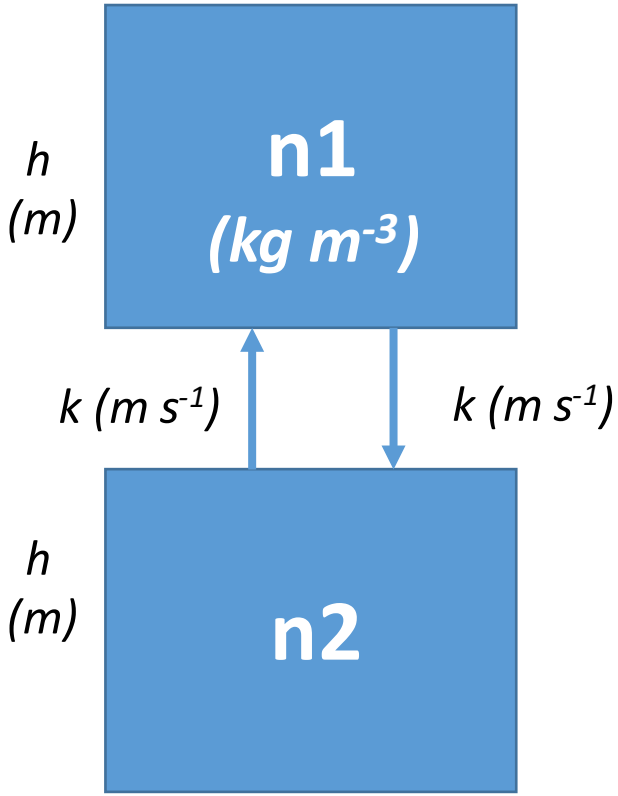
Markov chain representation of this model:

Box 1 -> Box 2 transition probability = k/h
Box 2 -> Box 1 transition probability = k/h
set k/h = 0.01 1% transition prob/time
proceed step-wise, stochastically (runif)



ar(x = N1) Coefficients: 0.9761 Order selected 1 (corr time = -1/log(.9761)= 41.3 (exact=50))

Our working example: a 2-box model mass transfer model, and equivalent Markov chain



$mass_i$ = h Area n_i

Example: turbulent exchange of trace gases

Model **pdf** **parameter**

Box 1 -> Box 2 transition probability = k/h
Box 2 -> Box 1 transition probability = k/h

observations

```
# R code for Markov chain representing the 2-box model
# (for reference)
k.h = 0.01                  # transition freq = 1/"lifetime"
n1 = 100; n2 = 0            # number of particles in each box, initial
N1 = rep(NA, 1000); N2 = rep(NA, 1000)
                            # number of particles in each box, in time
N1[1] = n1; N2[1] = n2
for(i in 2:1000) {
        if(n1 == 0) dn1 = 0 else dn1 = sum(runif(n1) < k.h)
        if(n2 == 0) dn2 = 0 else dn2 = sum(runif(n2) < k.h)
        # runif: uniform distribution random numbers between (0,1)
        n1 = n1-dn1+dn2         # new value in box 1
        n2 = n2-dn2+dn1         # in box 2
        N1[i] = n1
        N2[i] = n2
}
```

Fitting a model (e.g. a straight line ⇔ a model) to data:

Given a pair of covariates with a linear relationship,
we uncover and test that relationship
with a linear regression.

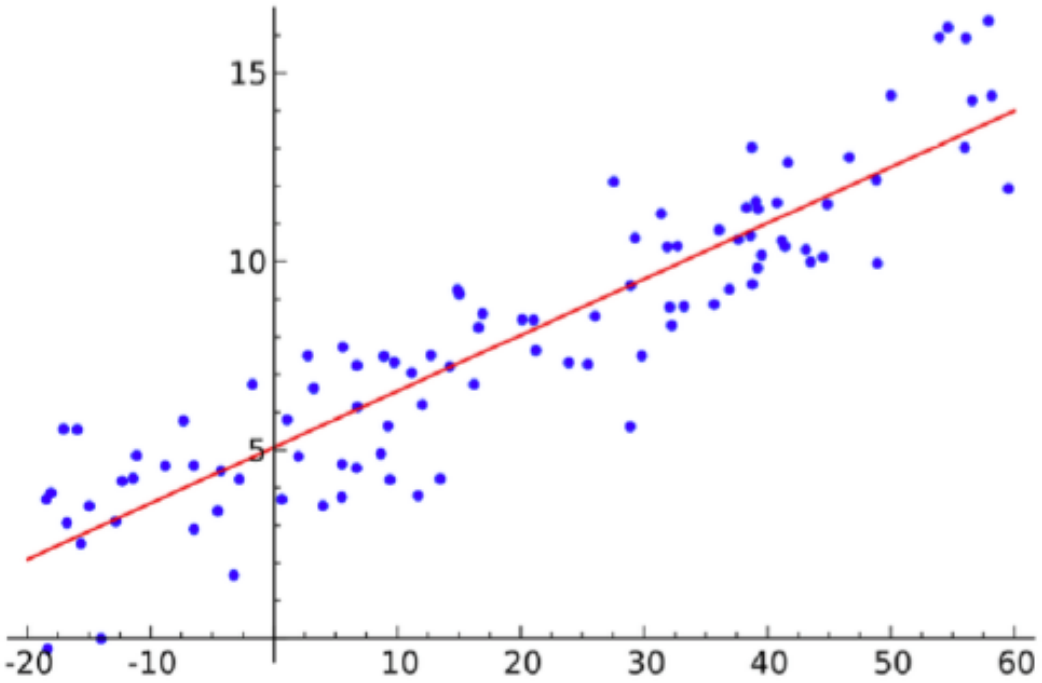
In R;
lm(y ~ x)

{ Pred , Obs }

$\{ x_i , y_i \}$

Model

$y = ax + b$



What are we really doing here?

Ordinary least squares as a maximum likelihood estimator

Likelihood is a term used for the probability of a set of parameters, given some data.

$$L(\theta|x) = P(x|\theta)$$

θ = parameters (a,b)
P = prob we get obs data x, given (a,b)

You can estimate parameters, given some data, by finding the parameters with the greatest likelihood of having produced the observations you obtained.

The likelihood provides you with a cost function or figure of merit to optimize.

Ordinary least squares as a maximum likelihood estimator

Assume that the errors in predicting the observations y using the covariates x and our model are normally distributed with the same variance σ.

σ is usually attributed to "measurement error", for y; no errors are attributed to x

Both usually false!!

Then the probability of the data is:

$$P \propto \prod_{i=1}^{N} \left[\exp\left(-\frac{1}{2} \left(\frac{y_i - y(x_i)}{\sigma} \right)^2 \right) \right]$$

← to maximize P

minimize sum of squared residuals

$$-\log(P) \propto \sum_{i=1}^{N} \frac{(y_i - y(x_i))^2}{2\sigma^2} \propto \sum_{i=1}^{N} (y_i - y(x_i))^2$$

And that's what motivated least squares

Ordinary Least Squares (OLS): "errors" all equal

Weighted Least squares (Chi-square): "Errors" may be different for different observations

OLS: $$\sum_{i=1}^{N} \frac{(y_i - y(x_i))^2}{2\sigma^2} \propto \sum_{i=1}^{N} (y_i - y(x_i))^2$$

χ^2 : $$-\log(P) \propto \sum_{i=1}^{N} \frac{(y_i - y(x_i))^2}{2\sigma_i^2}$$

So far we have not strayed from the familiar!

But there is (almost always) also error (unresolved variance) in the predictor (x) (our "symmetry").

Look at the relationship between the correlation coefficient ρ, and the OLS Slope (not equal!!!):

$\rho = \text{cor}(x,y) \cong r = \dfrac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2} \sqrt{\sum_i (y_i - \bar{y})^2}}$

symmetric

$\text{OLS Slope} = \rho\, \sigma_y/\sigma_x \cong r\, SD_y/SD_x$

asymmetric

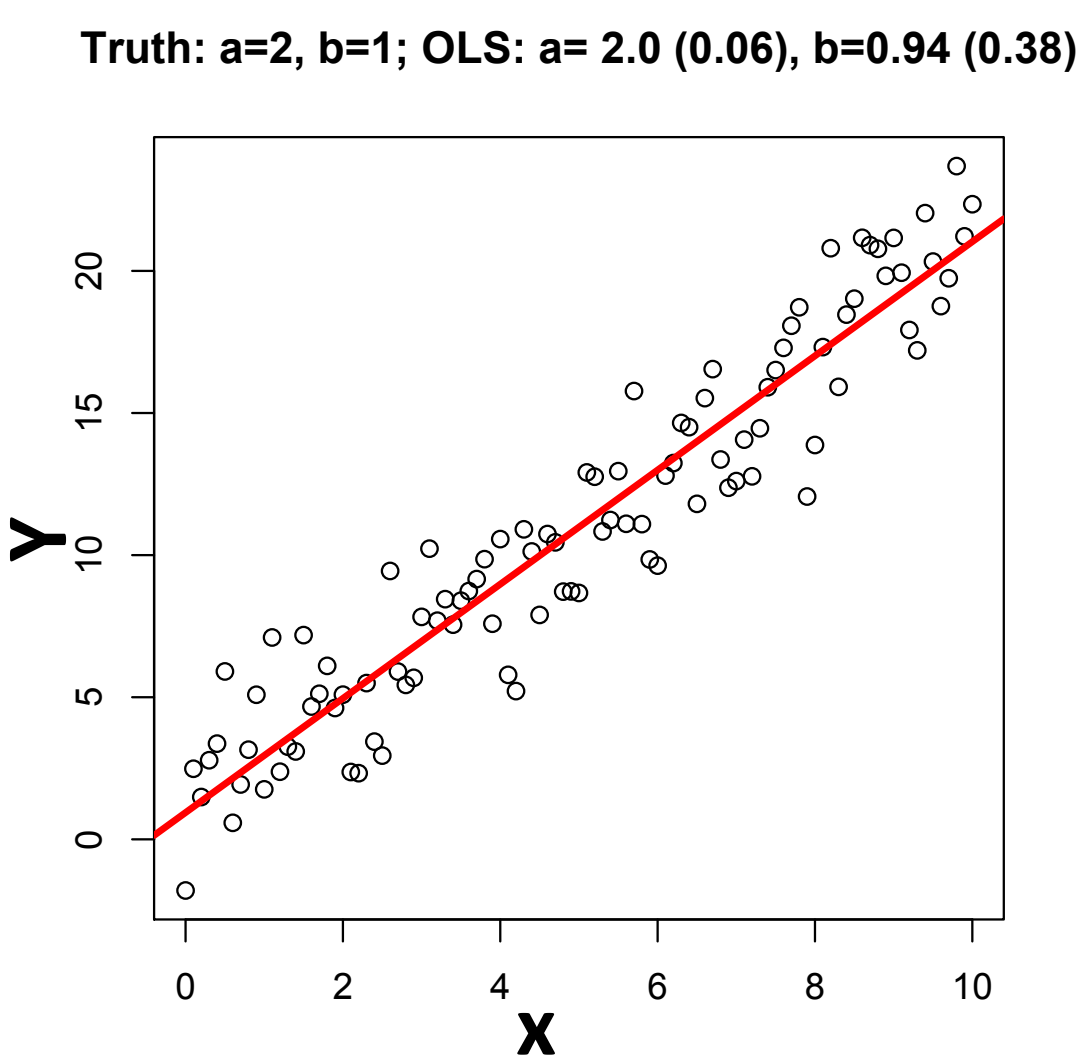
Why this is a YOOOGE problem

An example where OLS works very well

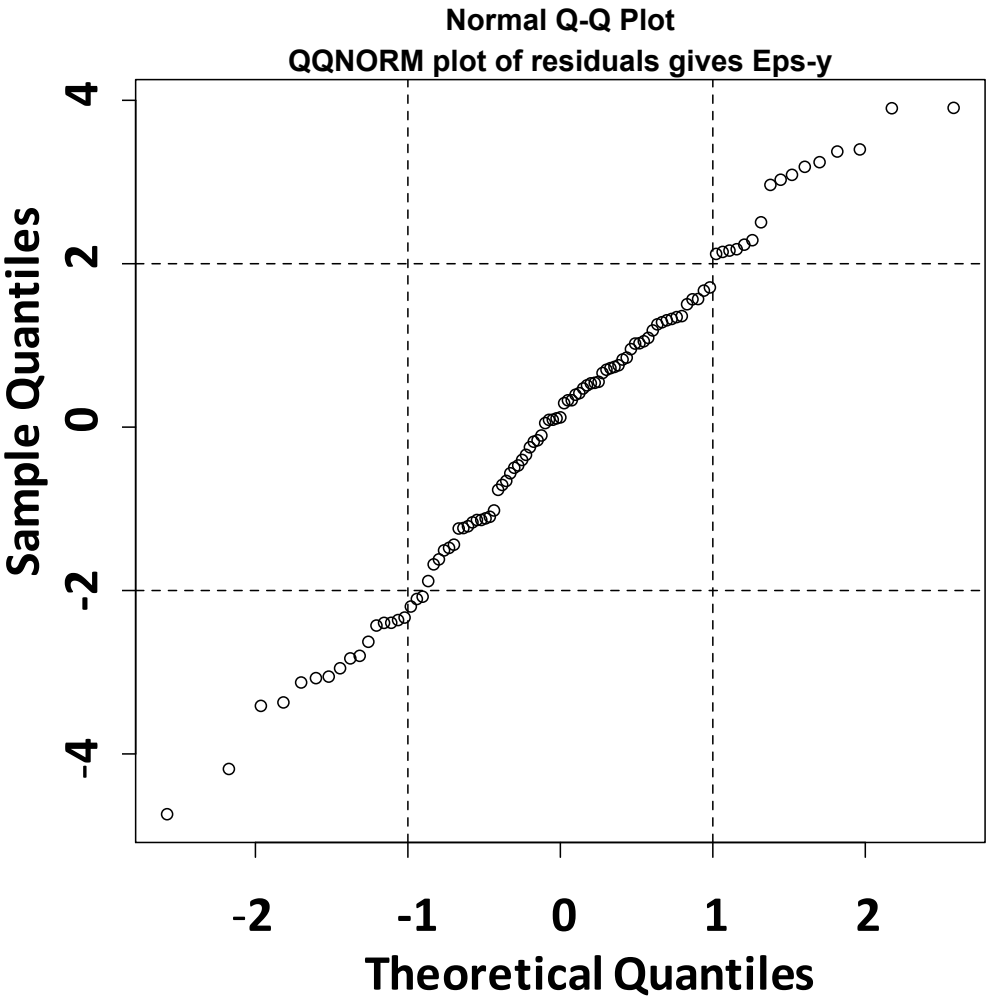
$y + \varepsilon_y = a + bx$

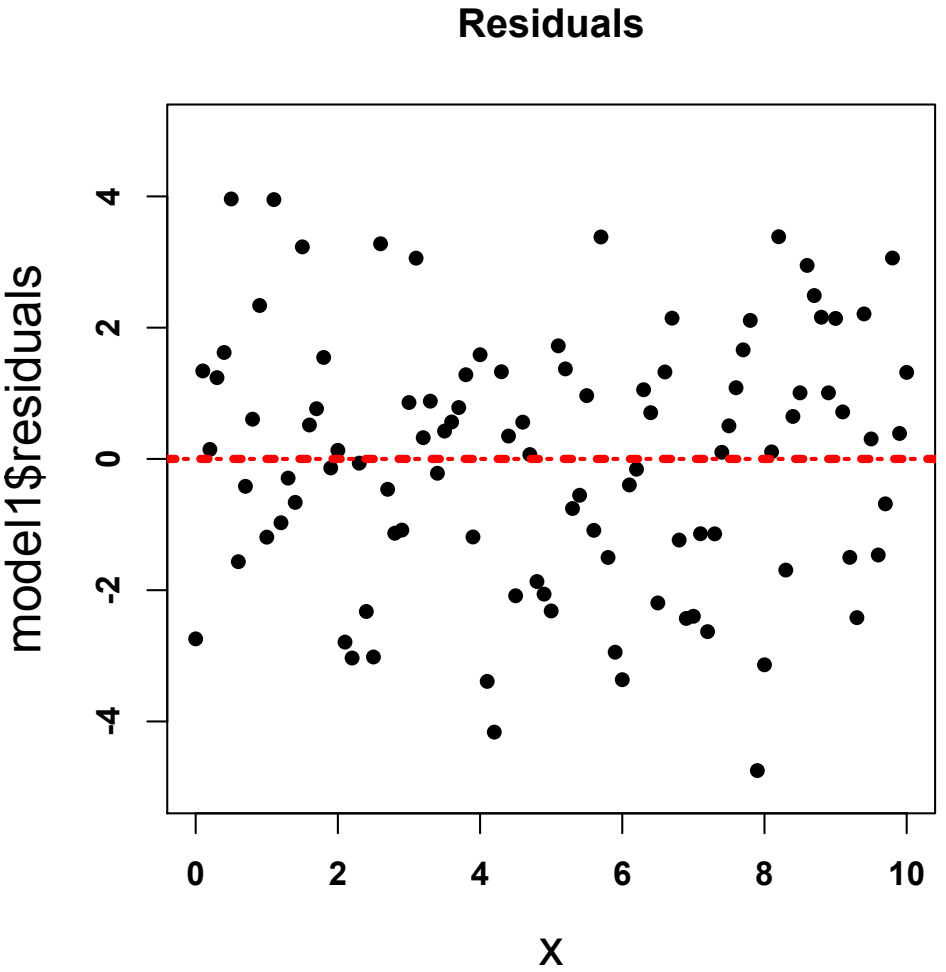
*x= 0:100 ; y= 1 +2*x + ε_y*

ε_y = rnorm(101,2)



… example where OLS works very well





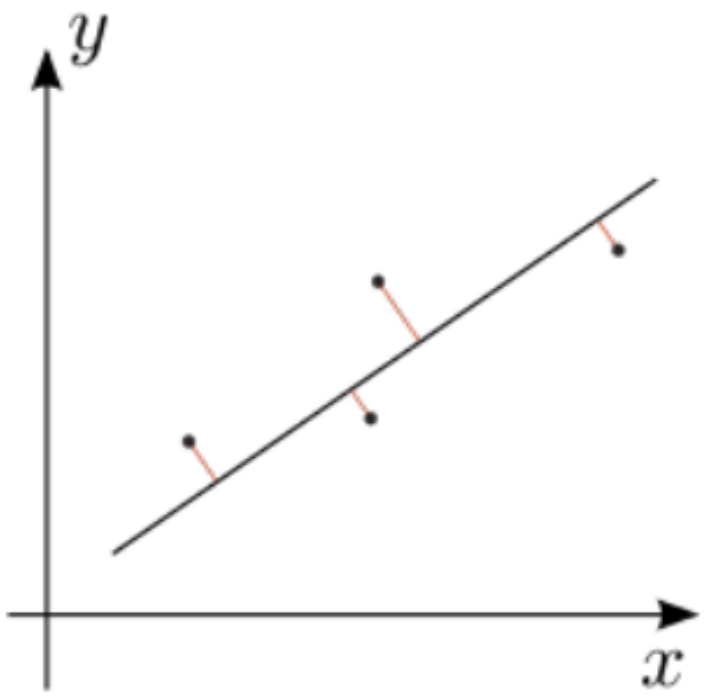
OLS discovers noise distribution ("error variance")!

OLS passes test for heteroscedacity (no trend in residuals)

Major Axis Regression

"Type II" regressions

Minimize the orthogonal distance from the points to the line
- "orthogonal least squares"



Available in the R package "lmodel2"

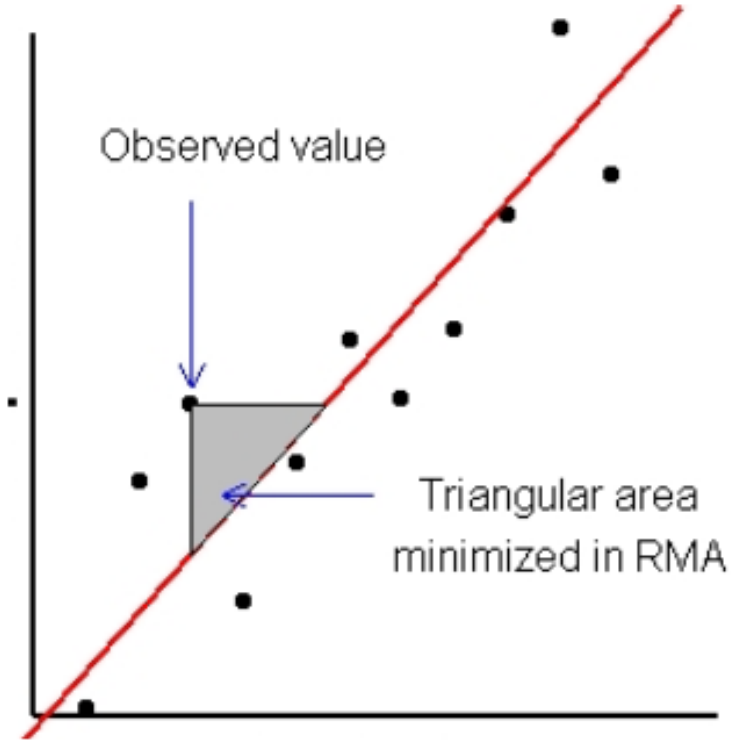
There is a symmetry
between the x and y.
Same units, same error.

There is no difference between
response and predictor.

Ranged Major Axis (RMA) and Standard Major Axis (SMA)

"Type II" regressions

Minimize the area under the triangle



Available in the R package "lmodel2"

Neither x nor y
should be considered a
response or predictor.

SMA recasts the data
into a bivariate normal distribution.

Generalized Chi-square regression

If experimental data are subject to measurement error not only in the y_i's, but also in the x_i's, then the task of fitting a straight-line model

$$y(x) = a + bx \tag{15.3.1}$$

is considerably harder. It is straightforward to write down the χ^2 merit function for this case,

$$\chi^2(a,b) = \sum_{i=1}^{N} \frac{(y_i - a - bx_i)^2}{\sigma_{y\,i}^2 + b^2\sigma_{x\,i}^2} \tag{15.3.2}$$

where $\sigma_{x\,i}$ and $\sigma_{y\,i}$ are, respectively, the x and y standard deviations for the ith point. The weighted sum of variances in the denominator of equation (15.3.2) can be understood both as the variance in the direction of the smallest χ^2 between each data point and the line with slope b, and also as the variance of the linear combination $y_i - a - bx_i$ of two random variables x_i and y_i,

$$\mathrm{Var}(y_i - a - bx_i) = \mathrm{Var}(y_i) + b^2\mathrm{Var}(x_i) = \sigma_{y\,i}^2 + b^2\sigma_{x\,i}^2 \equiv 1/w_i \tag{15.3.3}$$

The sum of the square of N random variables, each normalized by its variance, is thus χ^2-distributed.

We want to minimize equation (15.3.2) with respect to a and b. Unfortunately, the occurrence of b in the denominator of equation (15.3.2) makes the resulting equation for the slope $\partial\chi^2/\partial b = 0$ nonlinear. However, the corresponding condition for the intercept,

Press et al., *Numerical Recipes*

Generalized Chi-square regression:

y = a + bx

specified errors in both x and y, for each predictor-observation pair.

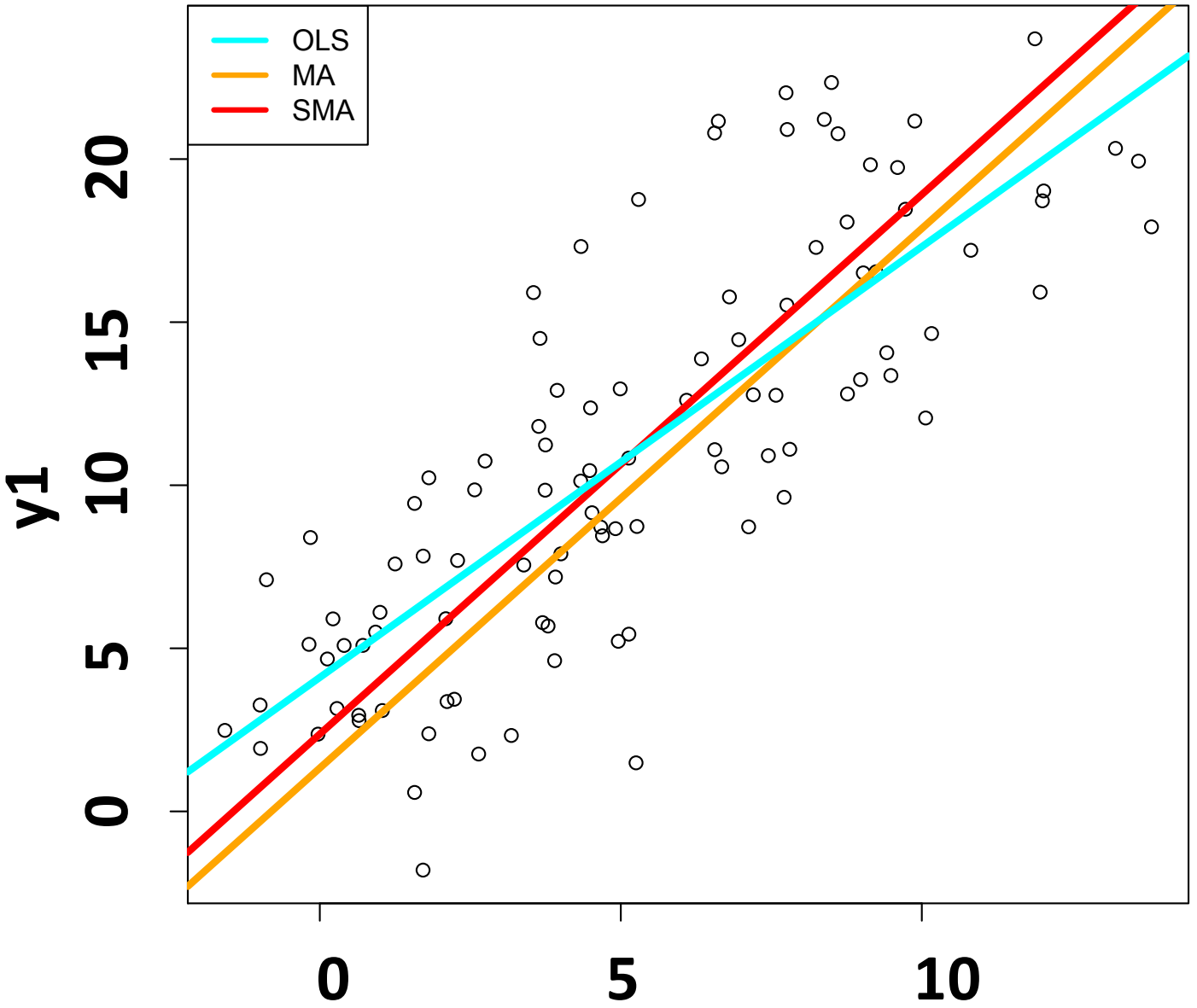
$$\chi^2(a,b) = \sum_{i=1}^{N} \frac{(y_i - a - bx_i)^2}{\sigma_{y\,i}^2 + b^2\sigma_{x\,i}^2}$$

Results in a non-linear optimization problem. Generalized to multivariate by York.

There are a variety of linear regressions that depend on your assumptions.

Method	Conditions	In R
OLS	Error on y is much greater than error on the x. Errors on y are iid.	lm(y~x)
WLS	Error on y is much greater than error on the x. Error on y have different weights.	lm(y~x, weights = …)
MA	Data is reasonably bivariate normal. x and y have same units and/or Error variances are close to the same.	lmodel2
RMA/ SMA	Data is reasonably bivariate normal. Error variance is roughly proportional to variance along axis.	lmodel2
York	You can estimate the error on x and y, ideally on individual x_i and y_i.	Yorkfit (web download)

$$y1 + \varepsilon_y = a + b\,x \quad ; x1 = x + \varepsilon_x$$



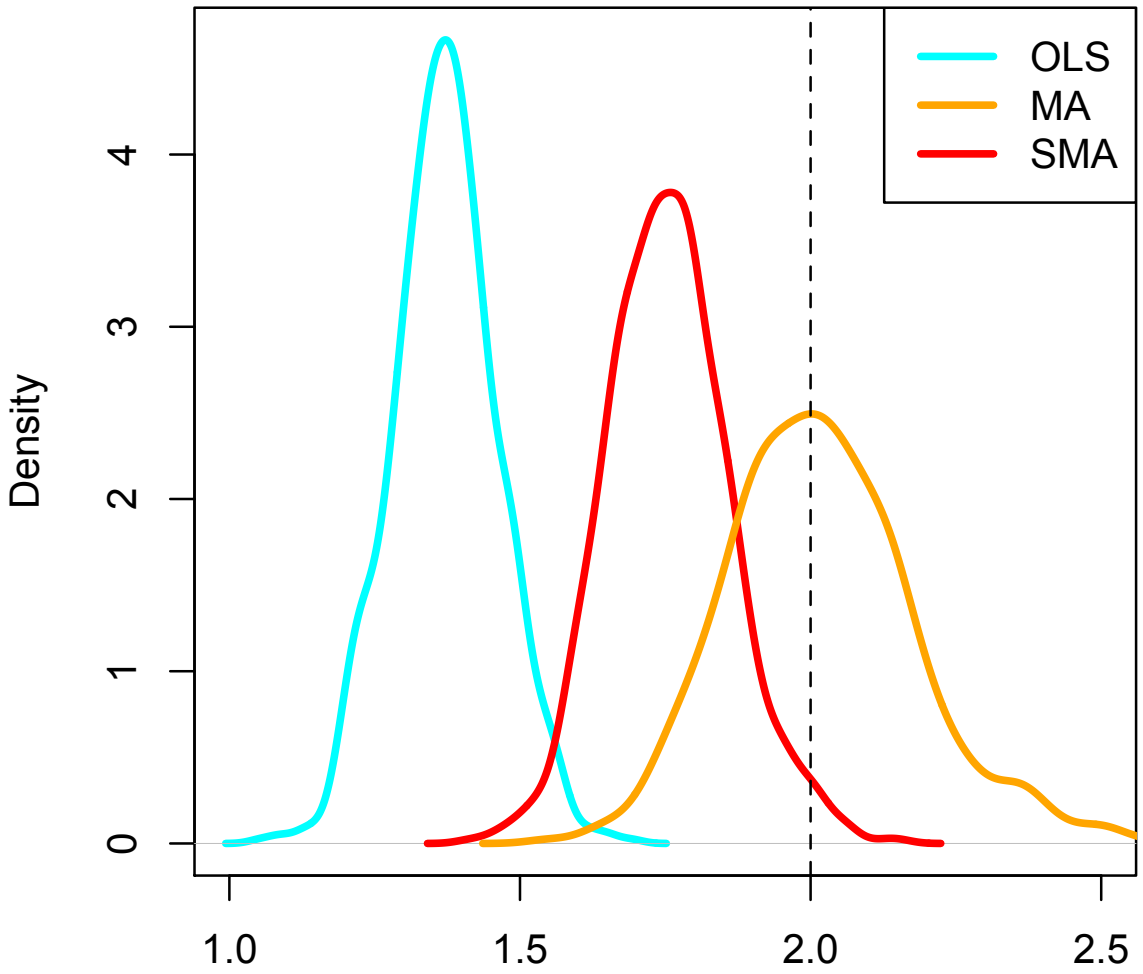
!!!!

OLS is biased towards underestimating the slope (…the "vooge" problem)

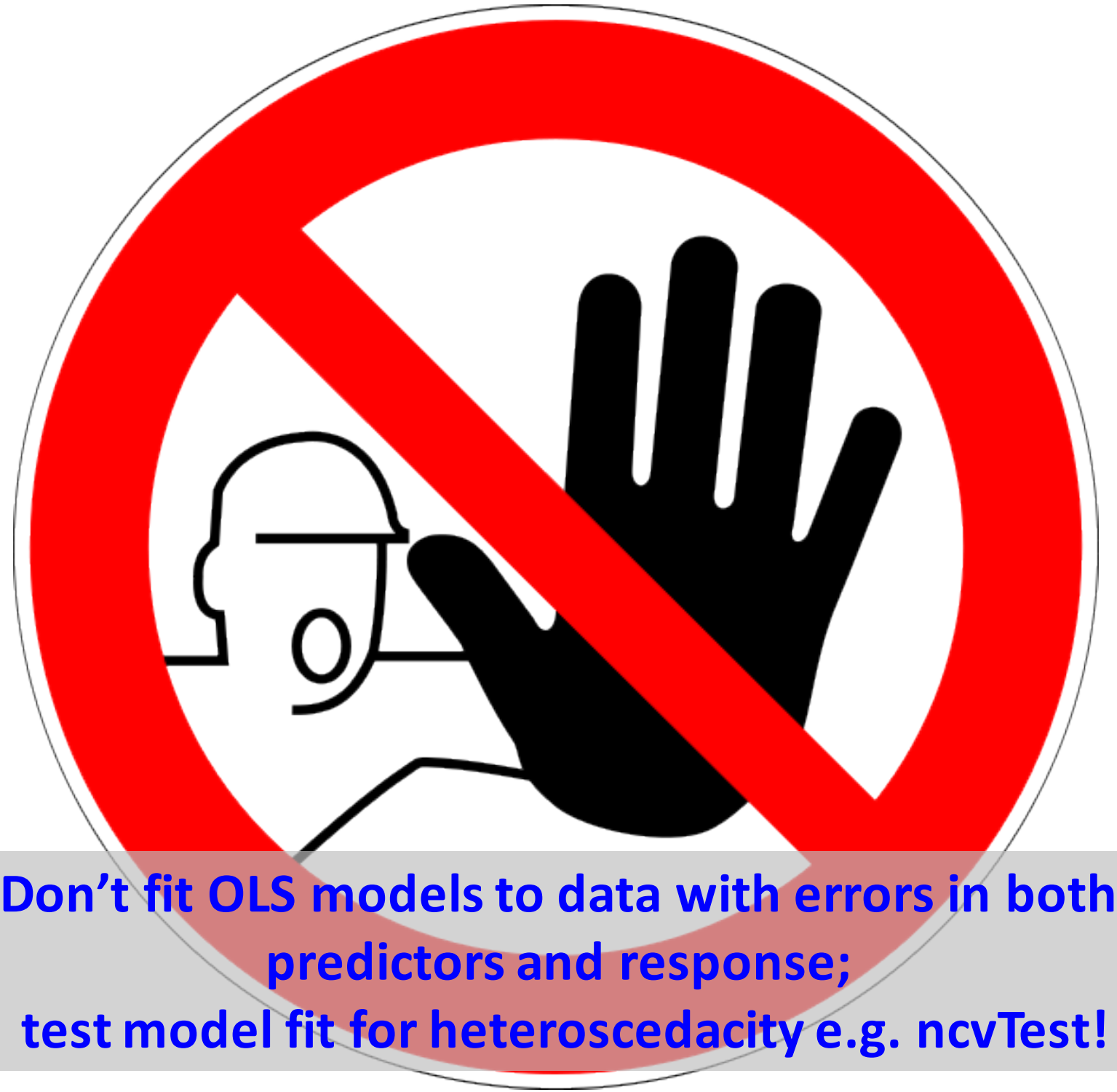
lmodel2(y1 ~ x1)

Slopes of Type I & II regressions for the model problem, resampled 1000 times (resampling errors)

MA is best, because the relationship between σ_x and σ_y is closest to MA assumptions



Regression slope Y1 on X1 – errors in both Y and X.



Don’t fit OLS models to data with errors in both predictors and response;
test model fit for heteroscedacity e.g. ncvTest!

3. Analyzing and Modeling Autocorrelated Data

- What is autocorrelation in a data set, and why is this characteristic so important in analyzing and modeling the data ?

- Brief introduction to time series analysis: autoregressive, moving average filtering of noise as a key element in time series.

- Variance inflation due to autocorrelation of data—”red shifted noise”.

- Using ARIMA models to extract underlying trends and signals from autocorrelated data.

3. First-order autoregressive (Markov) processes

Box, G. E. P., Jenkins, G. M., & Reinsel, G. C., "Time Series Analysis: Forecasting and Control" (Pearson, 1994).

Autoregressive Process (value at time t depends on previous value plus a random forcing):

Definition: $\mathbf{z_t = \varphi_1 z_{t-1} + a_t}$

z: observed data; t time (discrete, evenly spaced);
a random "shocks"; $\boldsymbol{\varphi_1}$ autoregressive parameter
(above is ar(1) process; note zero mean…)

Correlation function for lag k: $\rho_k = \phi_1 \rho_{k-1}$ ➔ $\rho_k = \phi_1^{\,k}$
(decays exponentially, osc if $\phi < 0$)

If $E[a_t^2] = \sigma_a^2$, $E[z_t^2] = \sigma_z^2 = \sigma_a^2/(1 - \rho_1\phi_1) = 1/(1 - \phi_1^2)$:
Variance *increased by* $1/(1 - \varphi_1^2)$ *compared to forcing !!!*

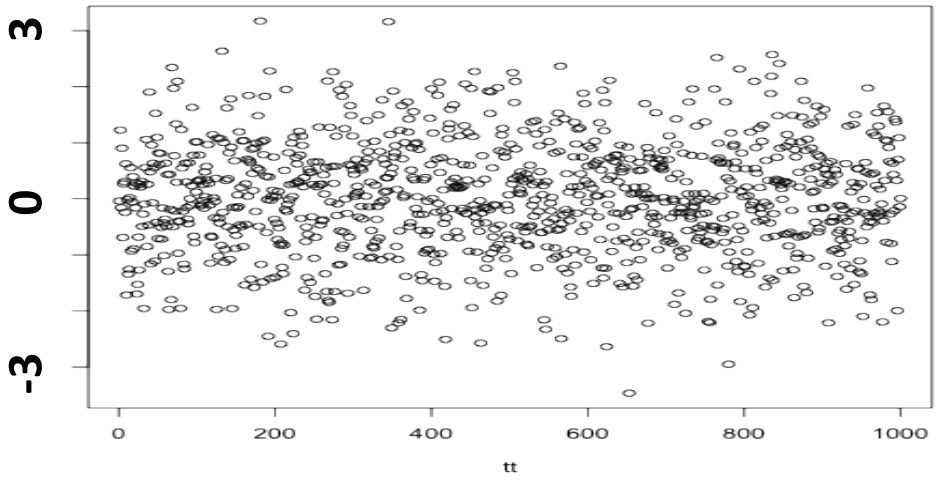
Random impulses with autocorrelation ➔ variance inflation

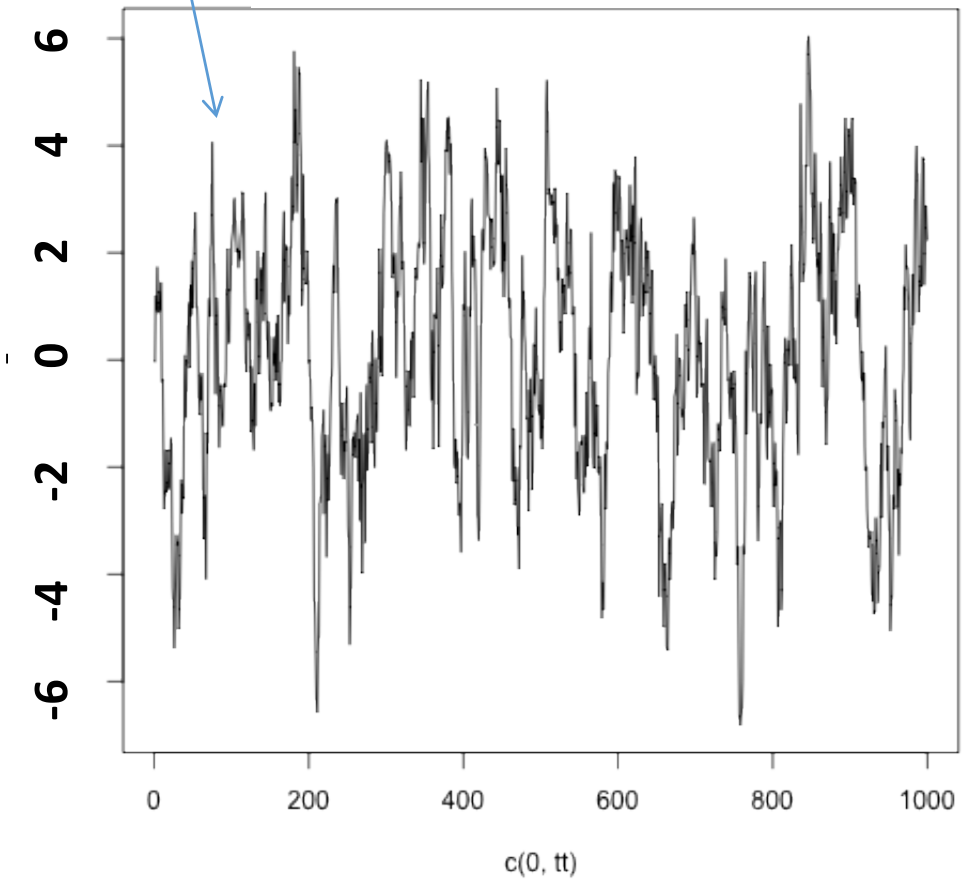
$\phi = 0.905$ $(\tau_{decay}=10)$

YY[i]=a[i]+ φ*YY[i-1]

$1/(1 - \phi^2) = 5.5$; var=4.97

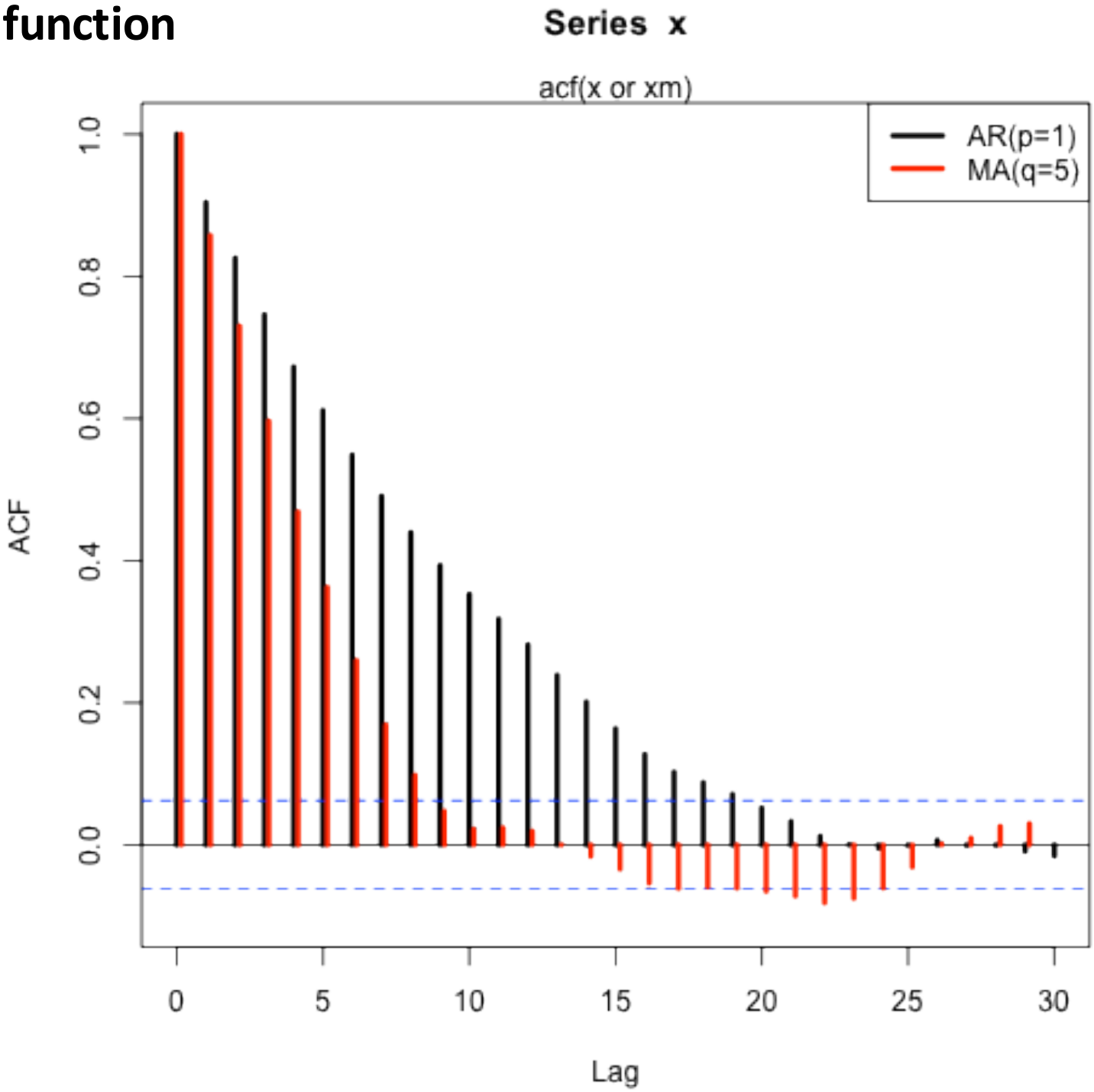
rnorm(1000); var= 1

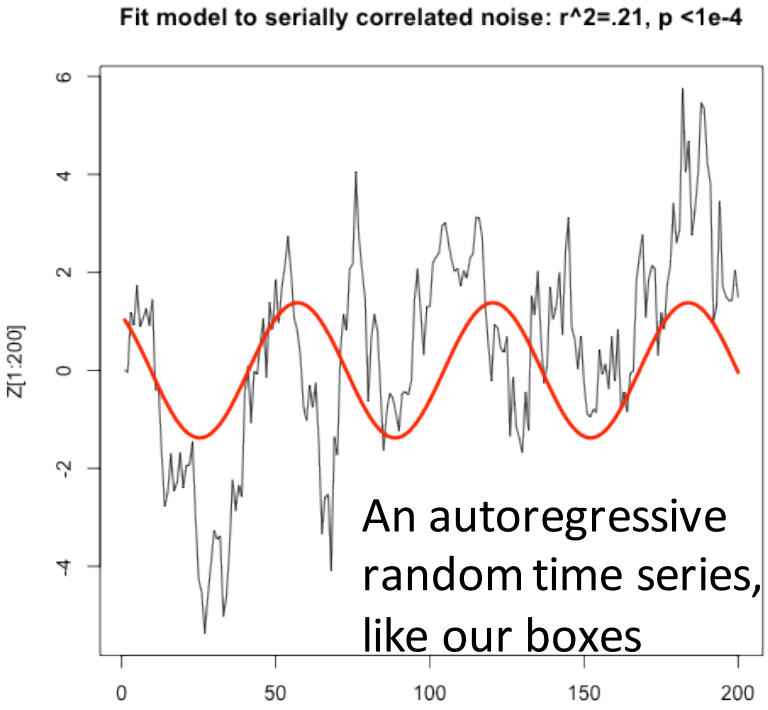


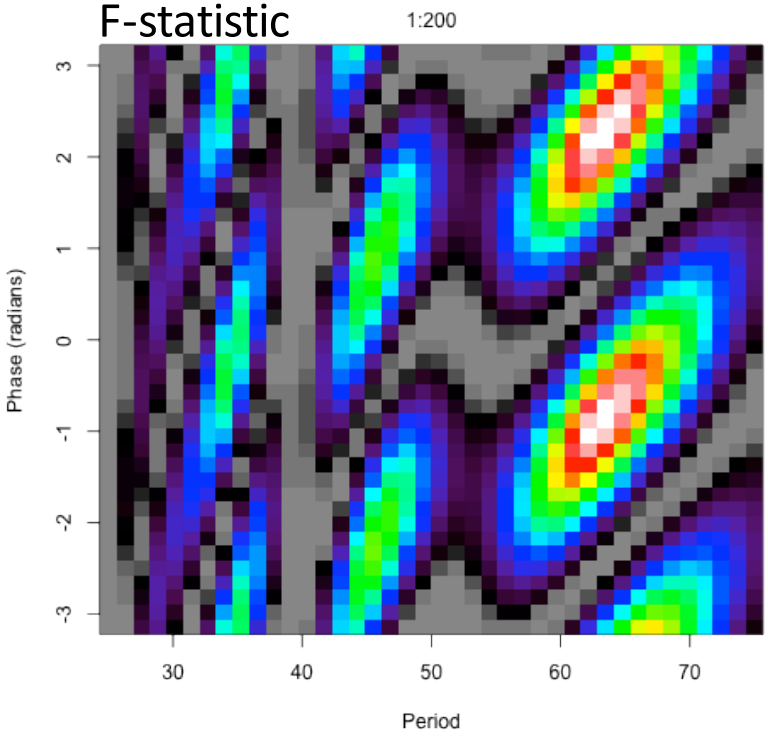


Autoregressive Ar(1) <first order…>

ACF function





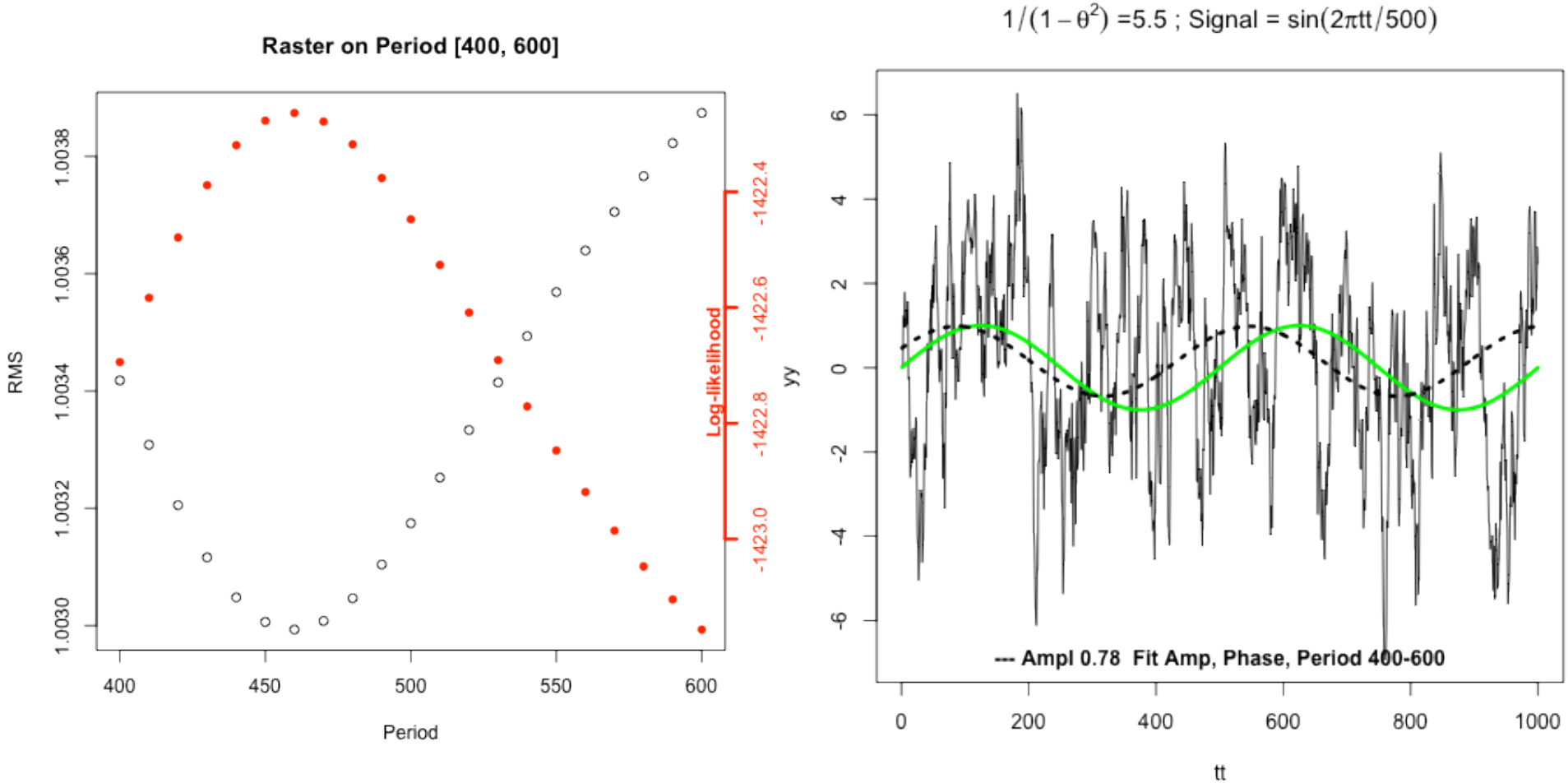



```
x33=sin((1:200)*2*pi/63.4-.94)
#ls.print(lsfit(x3,x[1:200]))
#Residual Standard Error=1.841 ; R-Square=0.2158
#F-statistic (df=1, 198)=54.4871; p-value=0
#            Estimate Std.Err t-value Pr(>|t|)
#Intercept   0.3883  0.1302  2.9821   0.0032
#X           -1.3795  0.1869 -7.3815   0.0000
arima(x = Z[1:200], order = c(1, 0, 0), xreg = x3)
#Coefficients:
```

The hazards of ar1() data:

- **There are actually fewer degrees of freedom than you think.**
- **The variance is inflated.**
- **The "redshifted noise" is actually random (in phase), but shows distinct structure associated with the decorrelation time. A very long time series needed to know if it is a signal**
- **Test data with ar() and arima()— efficiently estmate autoregressive coefficient and its significance.**

The arima(1,0,0) model determines signal with autocorrelated noise: it fits a basic model to both! ar() + OLS ….

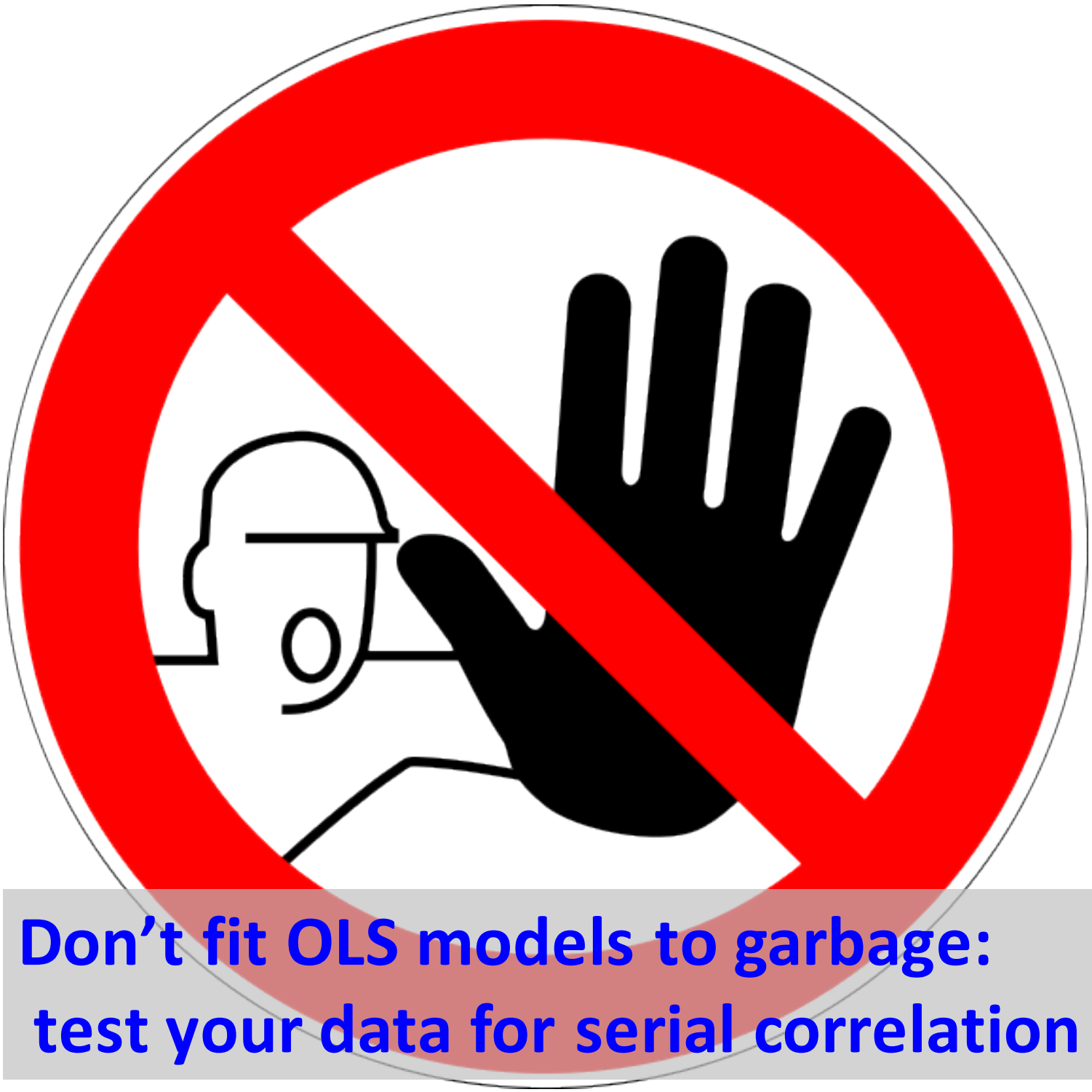


"Best" model: ar1=0.903, Ampl = 0.78 (s.e.0.4), sigma^2 (est. noise) = 1.001
Period=460, phase shift =-30.

	ar1	int	sin.t	cos.t
ar1	1.00000	0.00609	0.00952	0.00044
int	0.00609	1.00000	-0.06123	-0.10207
sin.t	0.00952	-0.06123	1.00000	-0.05594
cos.t	0.00044	-0.10207	-0.05594	1.00000

Parameter correlation matrix

Forecasting: predict.Arima()



Don’t fit OLS models to garbage:
test your data for serial correlation !

4. Estimating and presenting probability distributions and uncertainty (a.k.a. “errors”)

- Rigorously estimated errors are essential for data analysis
- The standard deviation **(σ)** measures variation of the sample.
- The *confidence interval* **(CI)** estimates the uncertainty of the model parameters.
- The confidence interval is often estimated using the “standard error” (from MLE) or the “t-test”. These assume errors are Gaussian.
- Complex inference (e.g. hierarcharal or sequential/chained models) require a bootstrap, or its big cousin, the Markov Chain/Monte Carlo approach.

What is a "confidence interval" ("CI") ?

The confidence interval is an estimate for the results that would be obtained from resampling (repeating an experiment).

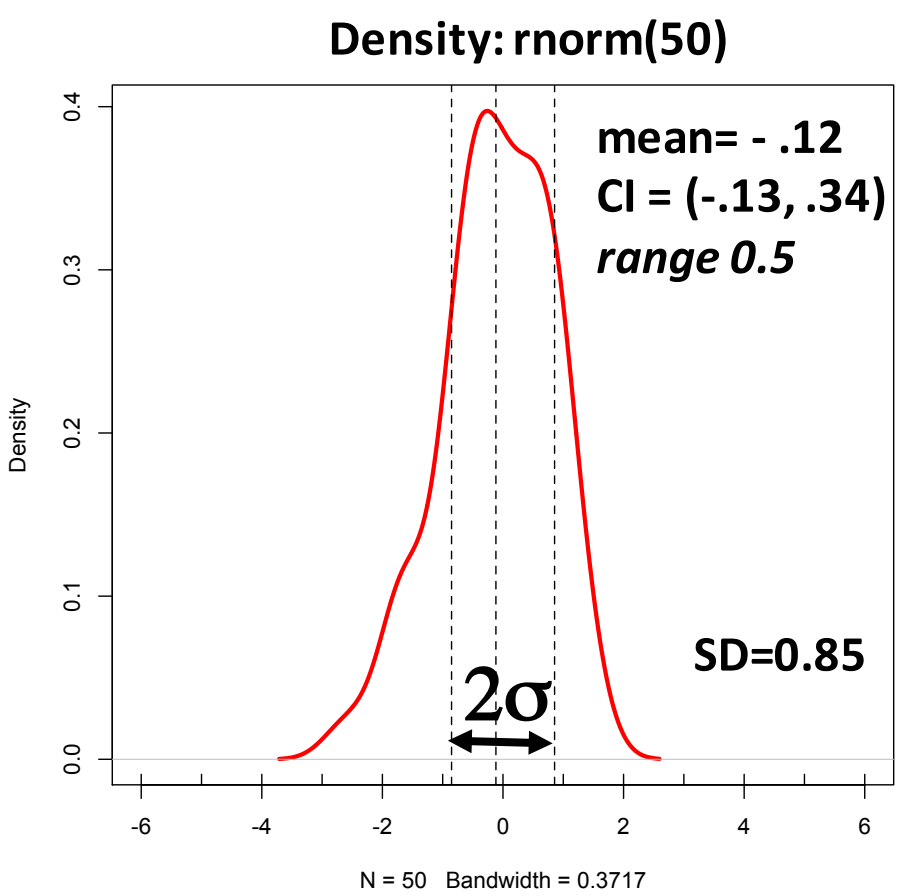
**If we could resample an experiment many times,
the x% (e.g. 95%) confidence interval encompasses x% (95%)
of the trial results.**

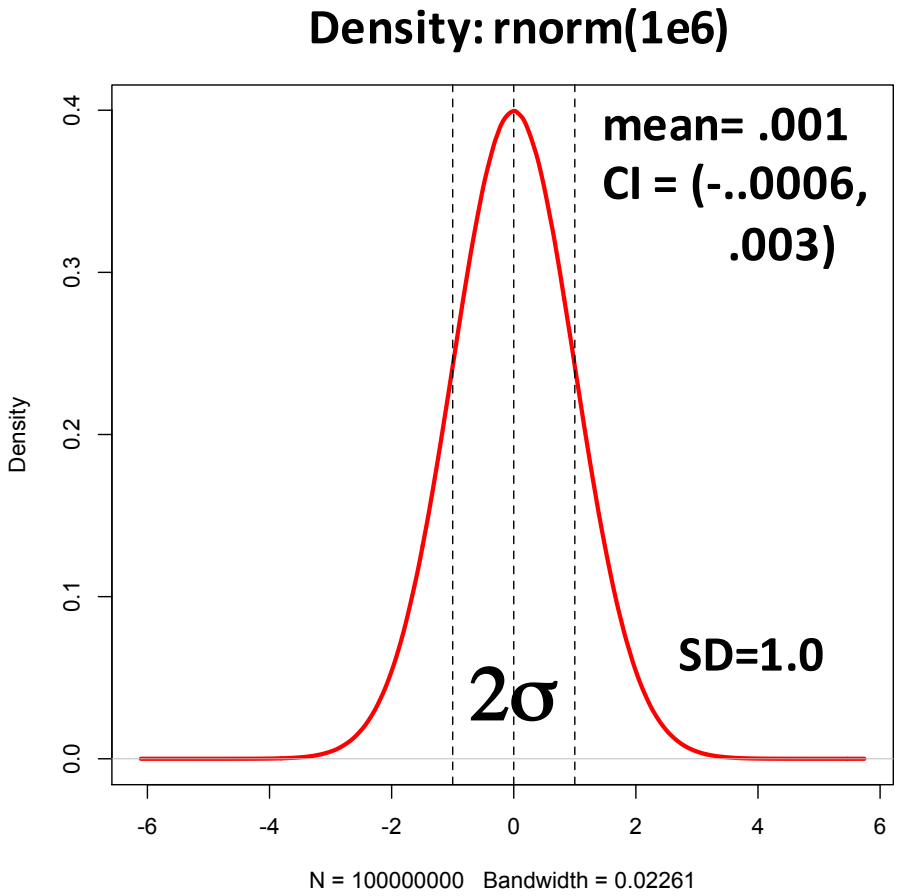
Example 1: I want to estimate the mean temperature over a geographic area at a particular time. I measure the temperature at N points across the domain, and take the average to obtain an estimate of the mean.

If I repeated this measurement many times, the 95% confidence interval is given the the 2.5 and 97.5 percentiles. I can only make an estimate of the true mean. Even worse, I can't resample the domain at a past time! I have only the N data points I first measured.

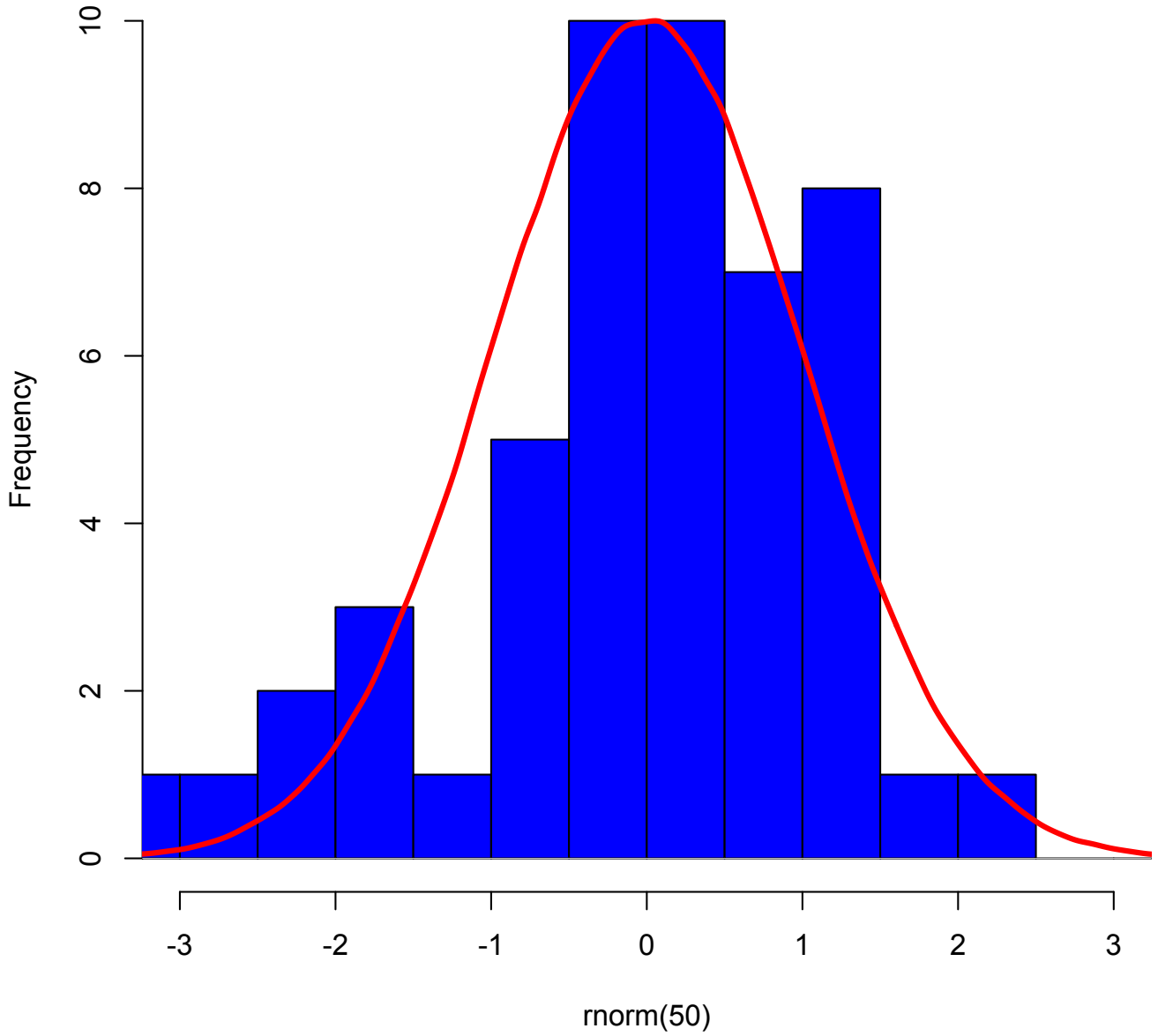
Standard deviation (σ) or variance (var)
vs
confidence interval (estimated error or uncertainty)

Standard deviation measures the width of the probability distribution of your sample: estimates of σ will be refined as the sample gets larger. Estimates of the **CI** get smaller as the sample gets larger.



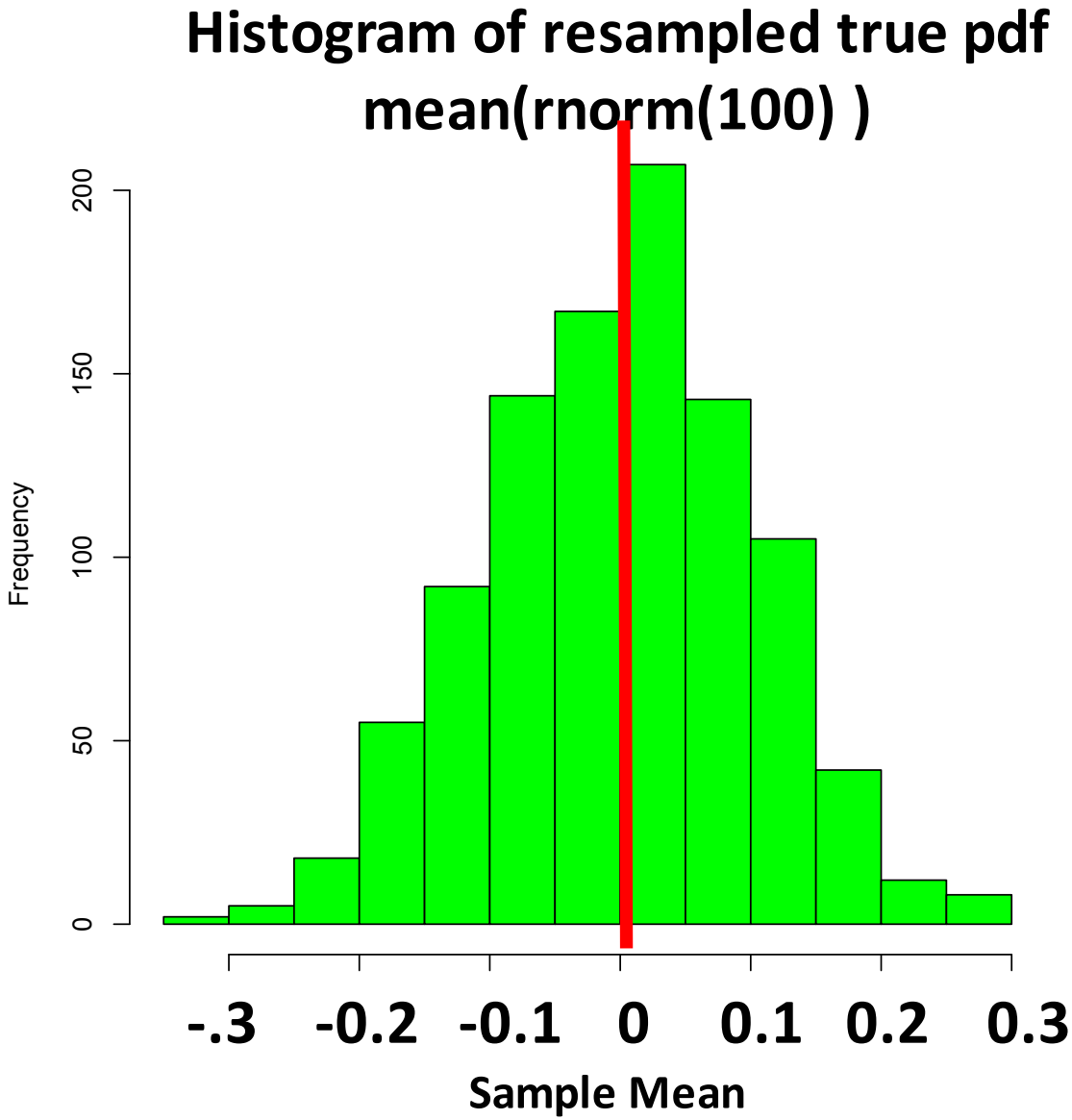


Histogram of 50 samples drawn from the normal distribution, compared to the (“true”) normal distribution



Example:
Estimate the mean of normally distributed data, N = 100 points, resampling the "true" Gaussian (Normal) pdf.

"True": mean =0
σ = 1 (std dev)



Resample 1000 times:

```
ZZ = rep(NA, 1000)
for(k in 1:1000){
    ZZ[k] = mean( rnorm(100) )
}
quantile(ZZ,probs=c(.025,.975))
```

2.5% , 97.5% ➔ -0.20, 0.20, ***CI range= 0.4***

Estimating errors using non-parametric resampling

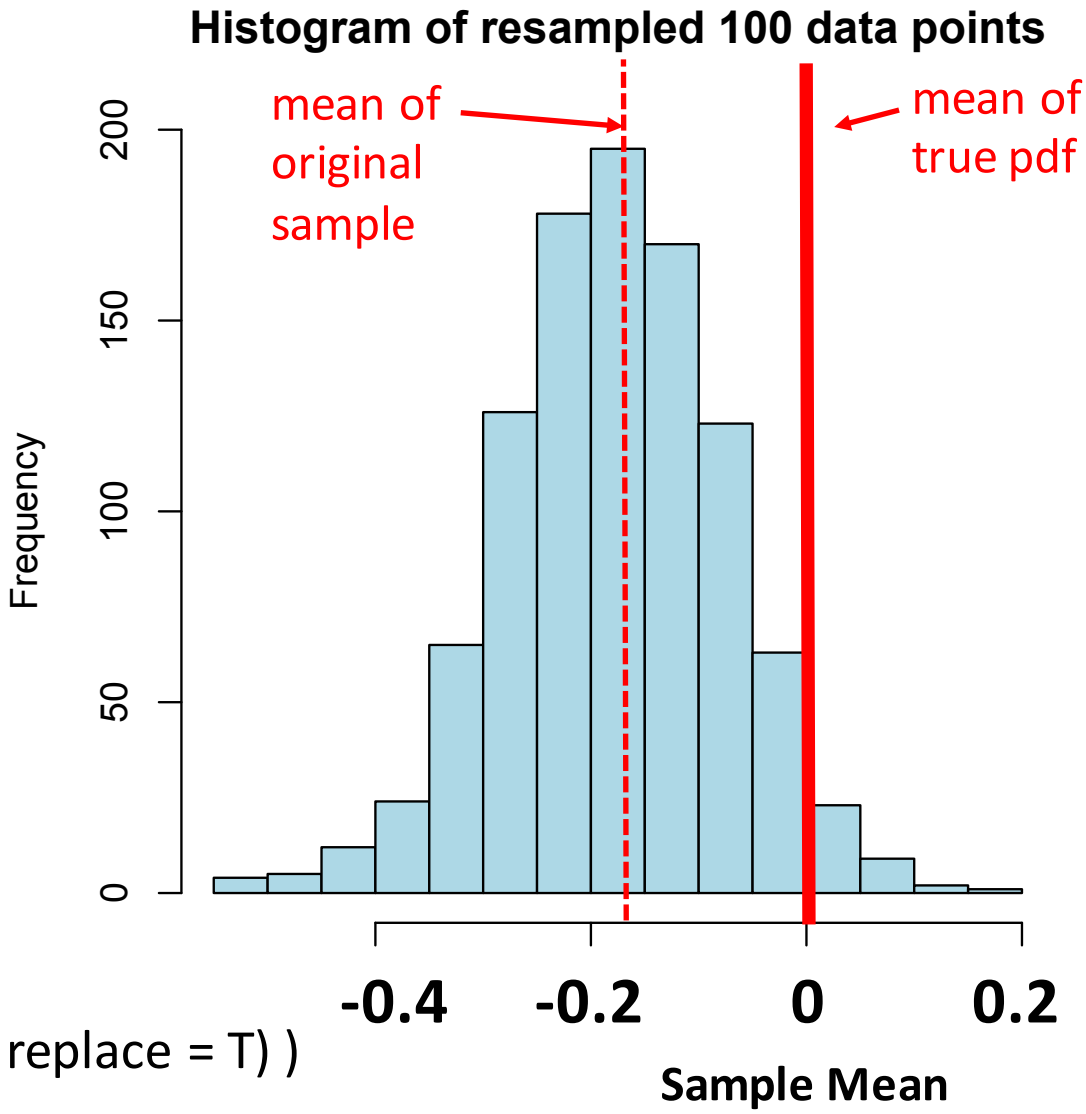
The ***bootstrap method*** ***resamples*** the original data set, i.e. if I can't repeat the measurements. We use its N data points to generate a large number of synthetic data sets, each also with N data points.

The procedure is simply to draw N data points at a time with replacement from each synthetic data set. Because of the replacement, we do not simply get back our original data set each time. We get sets in which a random fraction of the original points, on average $1/e \approx 37\%$, are replaced by duplicated original points.

Subject the original data and the synthetic data sets to the same estimation procedure (mean; MLE/Chi-square model inversion, …) to obtain parameter estimates.

Example:
Estimate the mean of normally distributed data, N = 100 points, using a bootstrap.

"True": mean =0
σ = 1 (std dev)



Resample 100 data 1000 times:

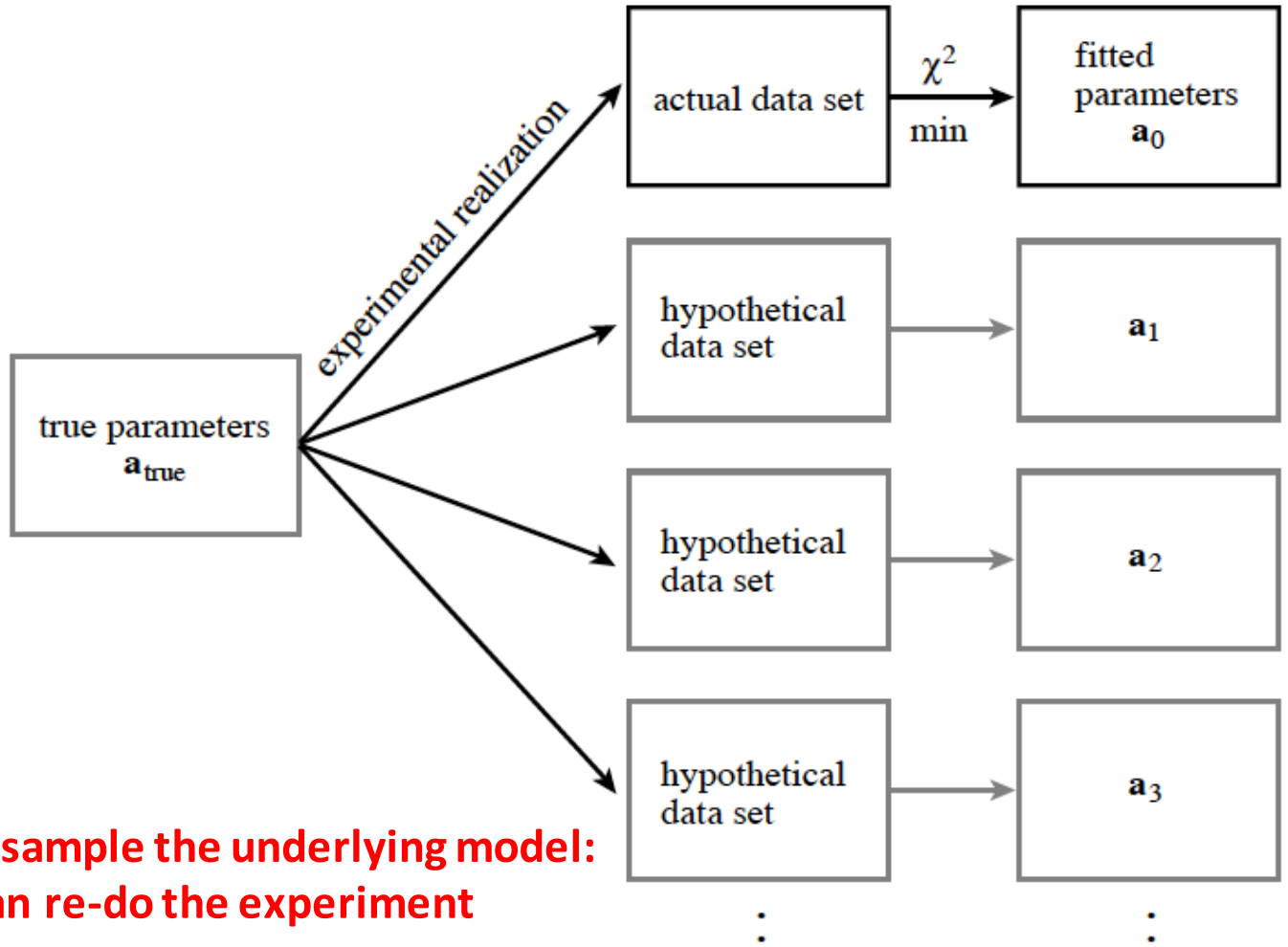
```
Zum = rnorm(100)
RR = rep(NA, 1000)
for(k in 1:1000){
    RR[k] = mean(sample(Zum, 100, replace = T) )
}
quantile(RR, probs = c(.025, .975))
```

2.5% , 97.5% ➔ -0.38 0.020 *range 0.40*

same range as sampling rnorm itself!

The set of estimated parameters will be distributed around the best estimate of the parameters from the original data (a(0)) in close approximation to the way that a(0) is distributed around a_{true} as in our simple example.

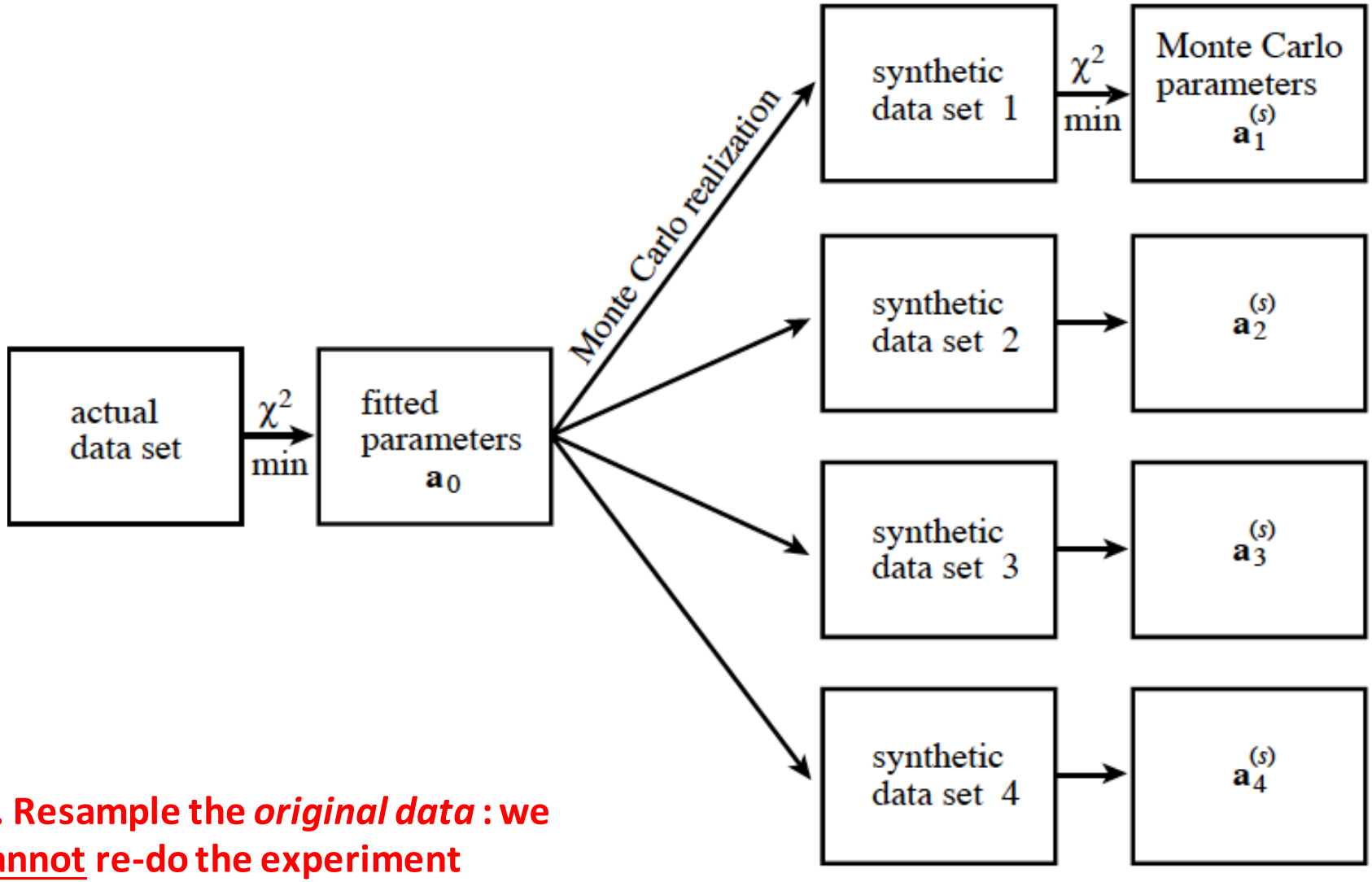
Markov chain – Monte Carlo concept to obtain the CI



A. Resample the underlying model: we can re-do the experiment

NUMERICAL RECIPES IN FORTRAN 77: THE ART OF SCIENTIFIC COMPUTING (ISBN 0-521-43064-X)

Figure 15.6.1. A statistical universe of data sets from an underlying model. True parameters $\mathbf{a}_{true}$ are realized in a data set, from which fitted (observed) parameters $\mathbf{a}_0$ are obtained. If the experiment were repeated many times, new data sets and new values of the fitted parameters would be obtained.



B. Resample the *original data* : we cannot re-do the experiment

Figure 15.6.2. Monte Carlo simulation of an experiment. The fitted parameters from an actual experiment are used as surrogates for the true parameters. Computer-generated random numbers are used to simulate many synthetic data sets. Each of these is analyzed to obtain its fitted parameters. The distribution of these fitted parameters around the (known) surrogate true parameters is thus studied.

Some critical things to take away from this lecture

1. **The basic hypotheses underlying statistical inference from data: the hypothetical "true" model or parameter value, and the equally hypothetical pure pdf that controls my sampling.**
2. **There is an underlying symmetry between "predictors" and "response": functional relationships, errors**
3. **Autocorrelation of (among) data and predictors is endemic in our fields and has a very strong influence on results**
4. **Use central values, standard deviation, uncertainty (standard errors), and confidence intervals correctly *and be explicit***
5. **The bootstrap or its big cousin, the Markov Chain Monte Carlo simulation, are incredibly useful tools for exploring the uncertainty structure of your problem. *They focus on repeated simulation and resampling of your data, or your complete data-model framework.***

Finis

First-order autoregressive (Markov) processes

Box, G. E. P., Jenkins, G. M., & Reinsel, G. C., "Time Series Analysis: Forecasting and Control" (Pearson, 1994**).**

Autoregressive Process (value at time t depends on previous value plus a random forcing):

Definition: $\mathbf{z_t = \varphi_1\ z_{t-1} + a_t}$

z: observed data; t time (discrete, evenly spaced); **a** random "shocks"; $\boldsymbol{\varphi}_1$ autoregressive weight parameter (above is ar(1) process; note zero mean…)

Moving Average Process (value at t depends a random forcing plus mean of previous):

Definition: $\mathbf{z_t = -\theta_1\ a_{t-1} + a_t}$

Both are special cases of stochastic processes with a linear filter:

Definition: $\mathbf{z_t = a_t} + \sum_{j=1}^{\infty} \mathbf{a_{t-j}}\ \psi_j$

z: observed data; t time (discrete, evenly spaced); a random "shocks"; φ_1 autoregressive weight parameter

Correlation function for lag k: $\rho_k = \phi_1\ \rho_{k-1}$ ➔ $\rho_k = \phi_1^{\,k}$ (decays exponentially, osc if ϕ <0)

If $E(a_t^2] = \sigma_a^2$, $E(z_t^2] = \sigma_z^2 = \sigma_a^2/(1 - \rho_1\phi_1) = 1/(1 - \phi_1^2)$: ***Variance*** *increased by* $1/(1 - \varphi_1^2)$

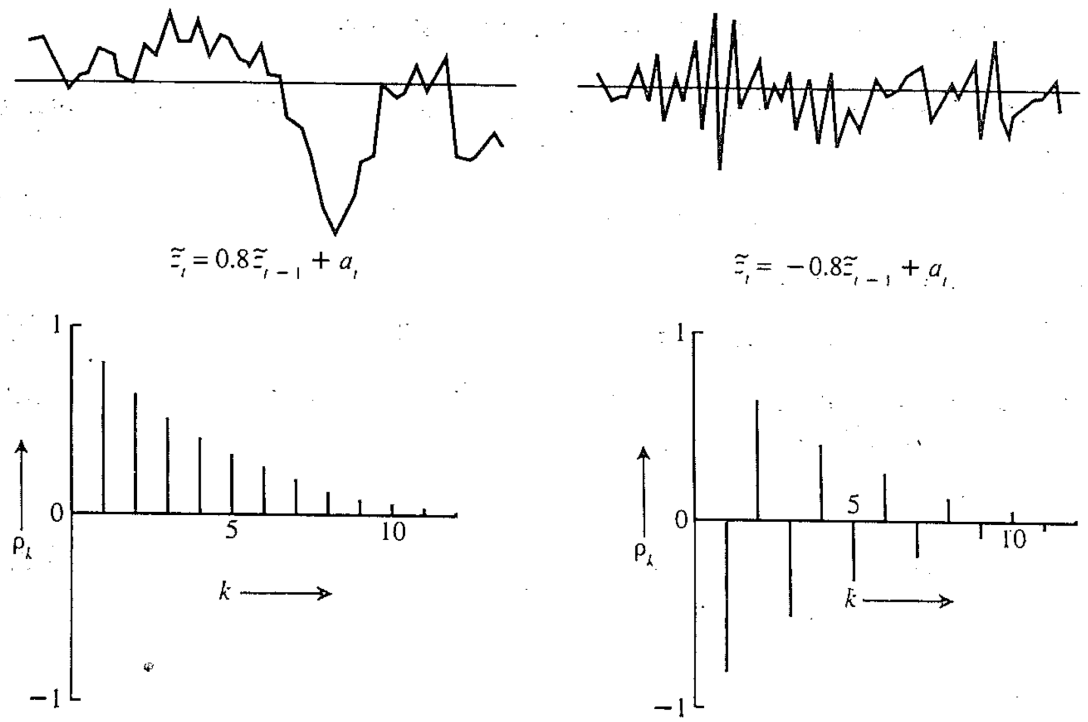


Figure 3.1 Realizations from first-order autoregressive processes and their corresponding theoretical autocorrelation functions and spectral density functions.