Chapter 5 and 6

DJM

21 February 2017

HW reminders

1. No code in the `pdf`.
2. If you put plots or tables in, you must talk about them.

   Rule of thumb: if you don't have anything good to say about a number, don't give the number (or plot) at all.
3. **MOST IMPORTANT** you must explain your results. Simply providing them is not likely to get you credit.
4. Look over the model solutions for the last assignments.

Project progress report

- Due **2 March at 11:59 pm** (just over 1 week from today)
- Your report should have 3 components:
  1. A list of teammate names and an explanation of what is interesting to you about this data.
  2. A short introductory paragraph introducing the data and describing some potential questions you might investigate.
  3. A lengthy exploratory data analysis.
- The third part is a big deal.
- You need to provide evidence that you have explored the data carefully and meaningfully.
- Code must be integrated.
- Think of this like HW 2.
- Just like with HW 2 and HW 3, much of what you do on the midterm will end up in the final report, so spend the time to do a good job.

Multiple datafiles

- The data come in one big file.
- This is not like data you have been working with before.
- I haven't cleaned it and made it easy to work with.
- You have to clean it and make it easy to work with.

Structure of the files

- Typically, you get data that looks like this:

```
##        y      x1     x2
## 1 -0.606 -0.6970 -0.643
## 2  0.252  0.0555  0.824
## 3 -0.748  0.8730 -0.182
## 4  0.890 -0.8150  2.230
## 5 -0.733 -0.5110 -1.270
```

- This time, your data looks like this:

```
load('../project/processedAttendance.Rdata')
head(attend)
```

```
##         Date       Day  Time Visit.Service.Category   Visit.Type
## 1 2011-02-01   Tuesday 12:00       CrossFit Classes CrossFit WOD
## 2 2011-02-01   Tuesday 12:00       CrossFit Classes CrossFit WOD
## 3 2011-02-01   Tuesday 19:15       CrossFit Classes CrossFit WOD
## 4 2011-02-01   Tuesday 19:15       CrossFit Classes CrossFit WOD
## 5 2011-02-01   Tuesday 19:15       CrossFit Classes CrossFit WOD
## 6 2011-02-02 Wednesday 17:15       CrossFit Classes CrossFit WOD
##                   Type                       Pricing.Option Exp..Date
## 1 6:00am CrossFit WOD                          Drop in Rate  2/1/2011
## 2 6:00am CrossFit WOD Monthly CrossFit Classes - Unlimited  3/1/2011
## 3 6:00am CrossFit WOD   Monthly CrossFit Classes - 3x/Week  2/4/2011
## 4 6:00am CrossFit WOD   Monthly CrossFit Classes - 3x/Week  2/3/2011
## 5 6:00am CrossFit WOD                          Drop in Rate  2/1/2011
## 6 6:00am CrossFit WOD Monthly CrossFit Classes - Unlimited  3/1/2011
##   Visits.Rem.    Staff    Visit.Location    Sale.Location Staff.Paid
## 1          10 Swinford Hoosier CrossFit Hoosier CrossFit         No
## 2        9980 Swinford Hoosier CrossFit Hoosier CrossFit         No
## 3        9983 Swinford Hoosier CrossFit     Online Store         No
## 4        9992 Swinford Hoosier CrossFit     Online Store         No
## 5           0 Swinford Hoosier CrossFit Hoosier CrossFit         No
## 6        9982 Swinford Hoosier CrossFit Hoosier CrossFit         No
##   Late.Cancel No.show Booking.Method Payment.Method Rev..per.Visit
## 1          No     Yes  Consumer Mode          Check          $8.75
## 2          No     Yes  Consumer Mode        Visa/MC          $5.00
## 3          No     Yes  Consumer Mode        Visa/MC          $6.18
## 4          No     Yes  Consumer Mode        Visa/MC         $13.13
## 5          No     Yes  Business Mode          Check          $8.75
## 6          No     Yes  Business Mode        Visa/MC          $4.44
##   Payment.Service.Category  IDs     class.date.time
## 1         CrossFit Classes  373 2011-02-01 12:00:00
## 2         CrossFit Classes  396 2011-02-01 12:00:00
## 3         CrossFit Classes  400 2011-02-01 19:15:00
## 4         CrossFit Classes  536 2011-02-01 19:15:00
## 5         CrossFit Classes 2257 2011-02-01 19:15:00
## 6         CrossFit Classes  175 2011-02-02 17:15:00
```

Structure of the files

- You need to decide what the "unit of observation" is.
  - Maybe it is an individual visit, then this data frame is "ok"
  - Maybe it is a class, then there are multiple records per class, how do you combine them?
  - Maybe it is a type of class, then there many records. Again how do you combine?

- **YOU** need to really think about this. You may (will, can, should) use all three to do some analyses.

Why Simulation?

- Up until now, when we do linear models, we used t-statistics, p-values, CIs
- These things are based on the sampling distribution of the estimators ($\hat{b}$) if the model is true and we don't do any model selection.
- What if we do model selection, use Kernels, think the model is wrong?
- None of those formulas work. And analogous formulas can be **impossible** (or painfully annoying) to derive.

Some simulation basics

```
set.seed(2017-02-21)
sample(1:10, replace=TRUE, prob=1:10/10)
```

```
##  [1] 10  5  7  2  2  9  3  2  7  8
```

```
sample(letters[1:10], replace=TRUE, prob=1:10/10)
```

```
##  [1] "j" "c" "e" "g" "h" "i" "f" "e" "j" "f"
```

```
sample(letters[1:10], replace=TRUE)
```

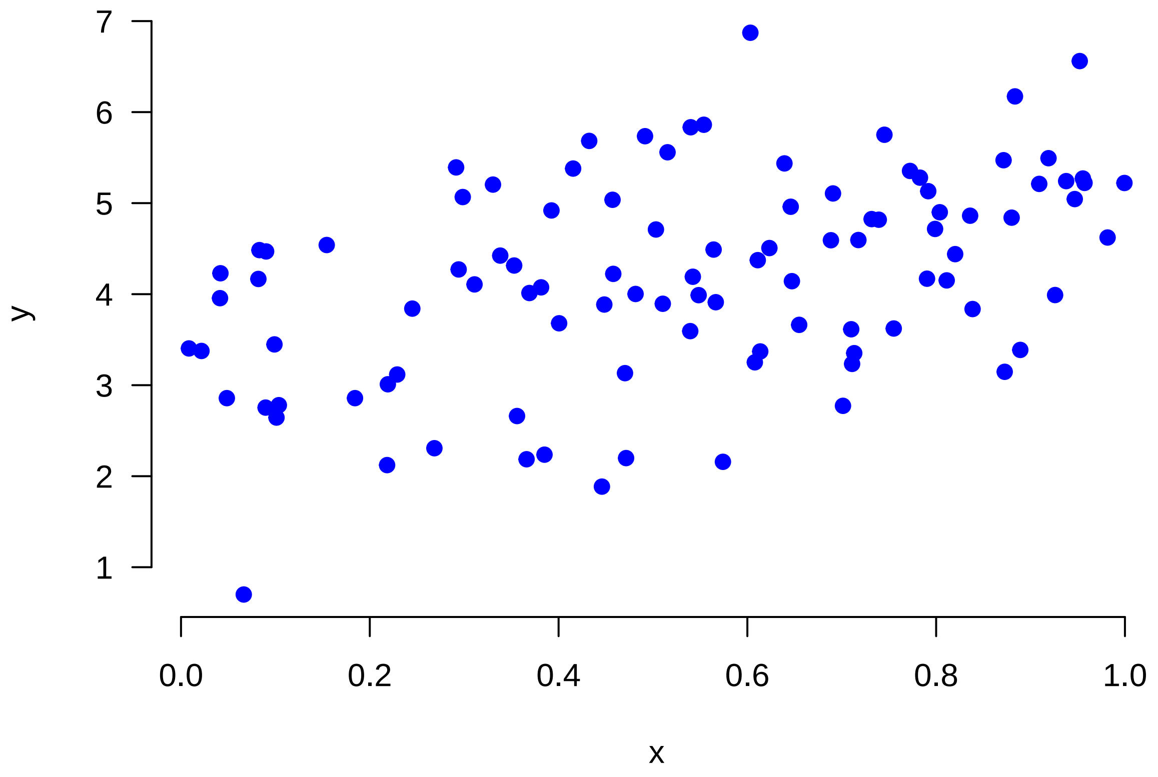
```
##  [1] "g" "g" "h" "c" "i" "d" "b" "a" "h" "f"
```

```
sample(letters[1:10])
```

```
##  [1] "g" "c" "f" "j" "b" "d" "a" "e" "i" "h"
```

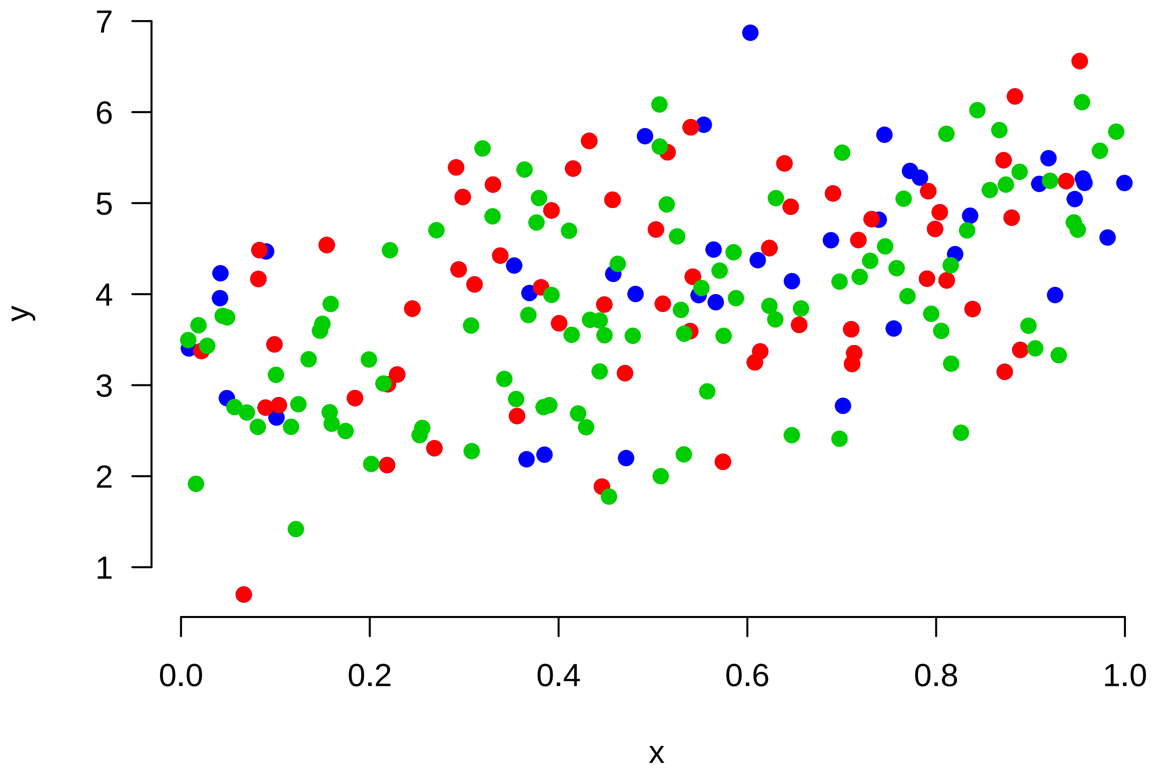
Resampling data

```
set.seed(2017-02-21)
n = 100; x = runif(n)
df = data.frame(x=x, y=3+2*x+rnorm(n))
plot(df, las=1, bty='n',pch=19, col=4)
```



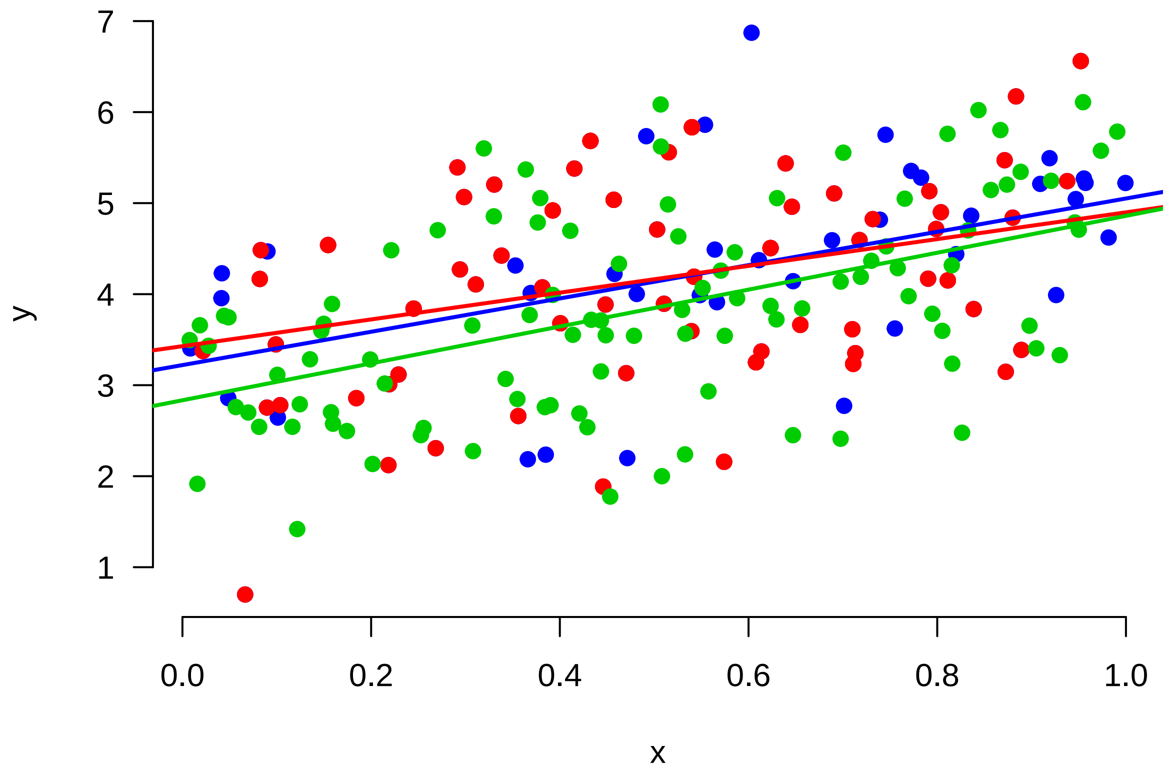
A sample (with replacement), and a new draw from the same distribution

```r
plot(df, las=1, bty='n',pch=19, col=4)
df2 = df[sample(1:n, replace=TRUE),]
xn = runif(n)
df3 = data.frame(x=xn, y=3+2*xn+rnorm(n))
points(df2, col=2, pch=19)
points(df3, col=3, pch=19)
```



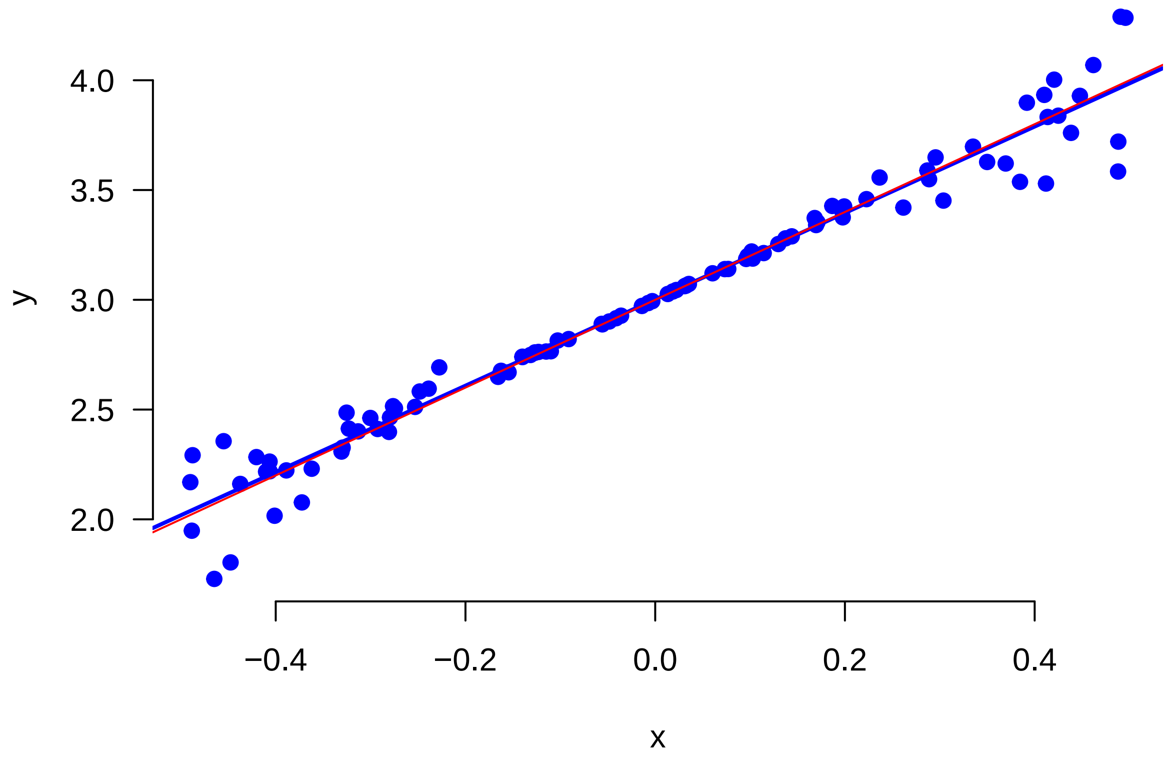
Add some lines

```
plot(df, las=1, bty='n',pch=19, col=4)
points(df2, col=2, pch=19)
points(df3, col=3, pch=19)
abline(lm(y~x, data=df), col=4, lwd=2)
abline(lm(y~x, data=df2), col=2, lwd=2)
abline(lm(y~x, data=df3), col=3, lwd=2)
```



Using simulations to check modelling assumptions

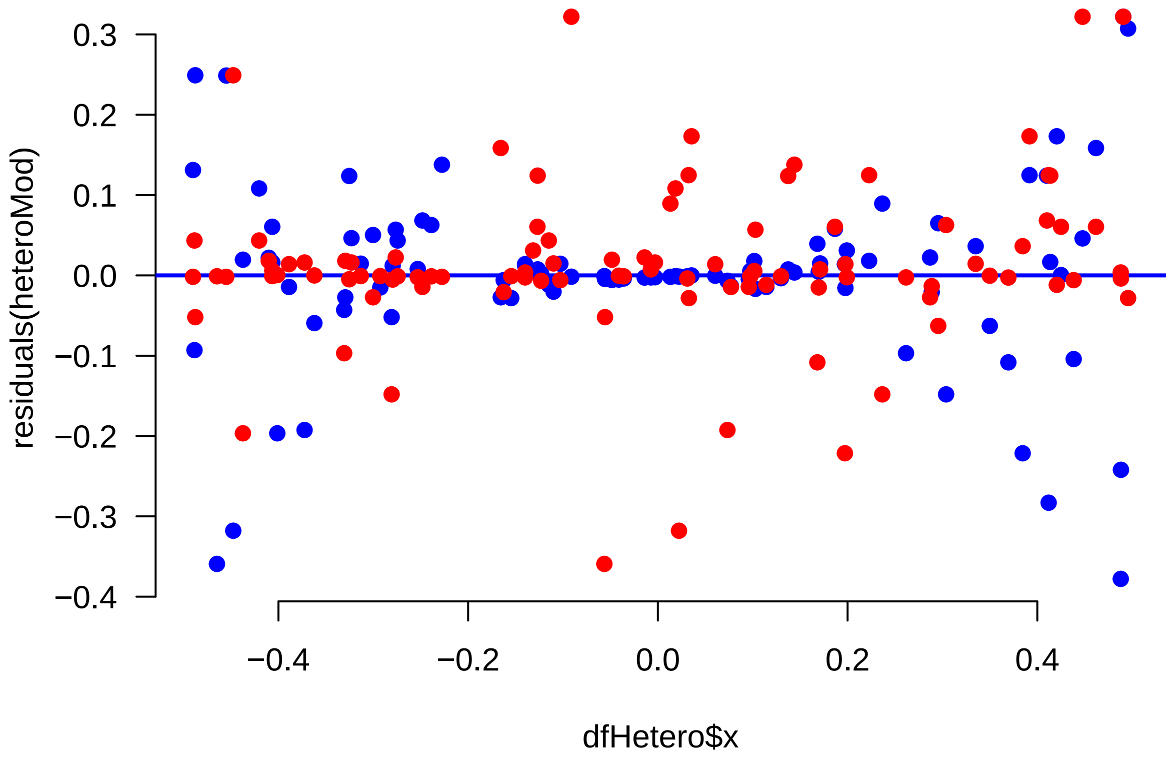
```r
x = runif(n) - 0.5; y = 3+2*x + rnorm(n)*x^2
dfHetero = data.frame(x=x, y=y)
plot(dfHetero, las=1, bty='n', pch=19, col=4)
abline(lm(y~x, data=dfHetero), col=4, lwd=2)
abline(a=3, b=2, col=2)
```



If the noise is homoskedastic...

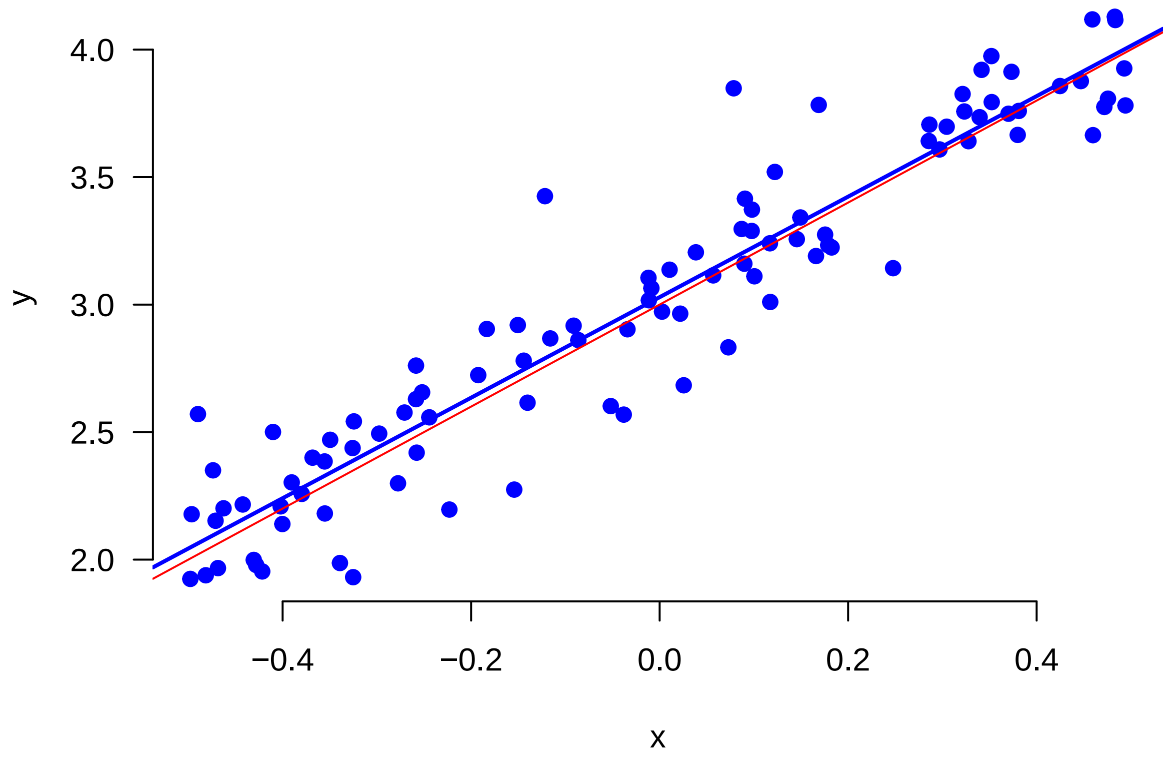
- The red and blue points should have the same distribution

```r
heteroMod = lm(y~x, data=dfHetero)
plot(dfHetero$x, residuals(heteroMod), las=1, bty='n', pch=19, col=4)
abline(h=0, col=4, lwd=2)
points(dfHetero$x, residuals(heteroMod)[sample(1:n, replace=TRUE)], col=2, pch=19)
```



That one was easy

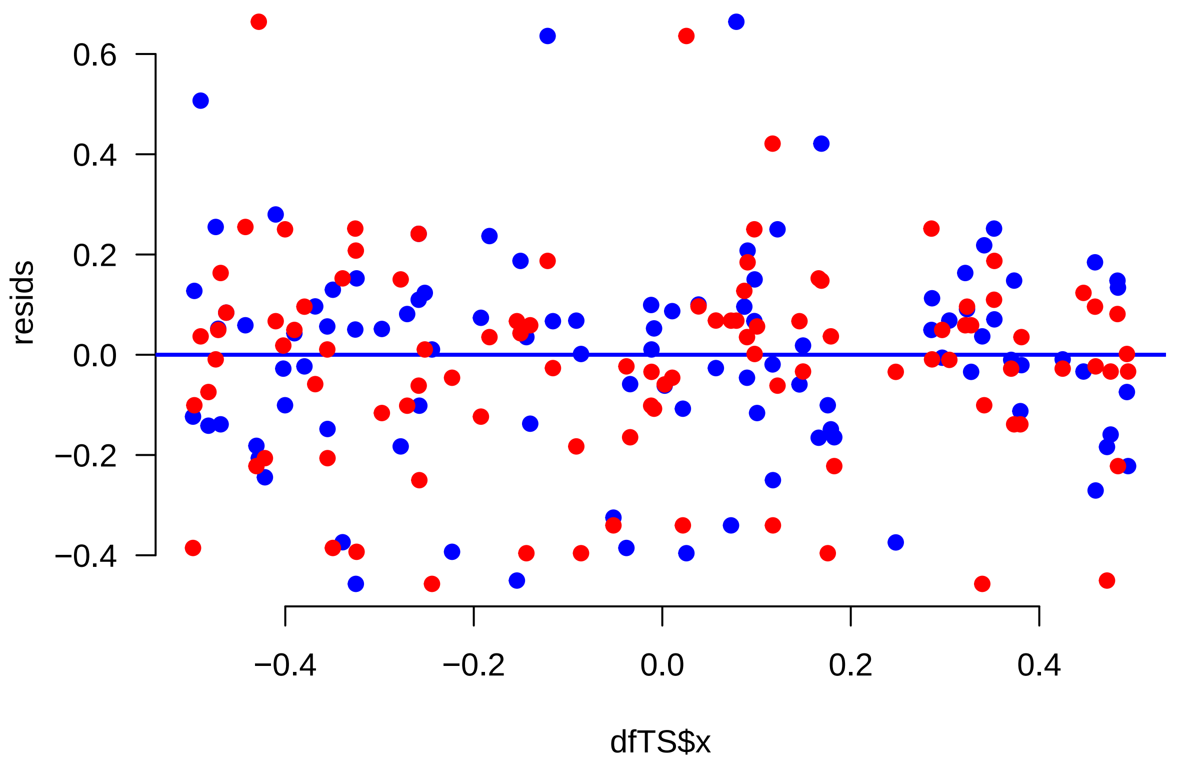
```
x = runif(n)-0.5
y = 3+2*x + c(arima.sim(list(ar=.8), n, rand.gen = function(n) 0.1* rt(n, df=5)))
dfTS = data.frame(x=x, y=y)
plot(dfTS, las=1, bty='n', pch=19, col=4)
abline(lm(y~x, data=dfTS), col=4, lwd=2)
abline(a=3, b=2, col=2)
```



If the noise is homoskedastic. . .

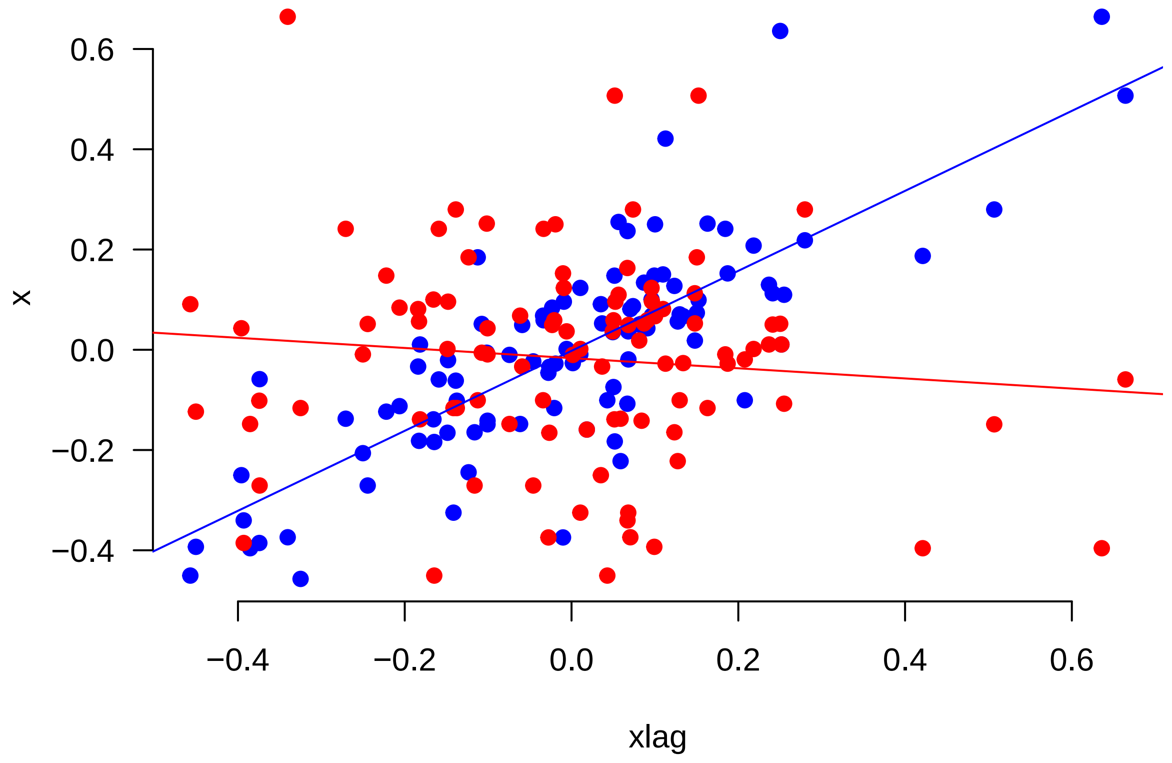
- The red and blue points should have the same distribution

```r
tsMod = lm(y~x, data=dfTS)
resids = residuals(tsMod)
plot(dfTS$x, resids, las=1, bty='n', pch=19, col=4)
abline(h=0, col=4, lwd=2)
points(dfTS$x, resids[sample(1:n, replace=TRUE)], col=2, pch=19)
```



But. . .

```
lag.resids = data.frame(xlag = resids[-n], x = resids[-1])
plot(lag.resids, las=1, bty='n', pch=19, col=4)
resamp.lag.resids = data.frame(xlag=resids[-n],
                          x=resids[sample.int(nrow(lag.resids), replace=TRUE)])
points(resamp.lag.resids, col=2, pch=19)
abline(lm(x~xlag, data=lag.resids), col=4)
abline(lm(x~xlag, data=resamp.lag.resids), col=2)
```



Another useful command

```r
sample.int(10)
```

```
##  [1]  6  4  3  1  2  8  9  5  7 10
```

What's the deal with this Bootstrap?

What's the deal with this Bootstrap?

- Suppose I want to estimate something and get a CI.
- But I don't know how to calculate the CI (or maybe I do, but it's hard)
- Then what?

Example 1

- Let $X_i \sim \chi^2_4$.
- I know if I estimate the mean with $\bar{X}$, then by the CLT (if n is big),

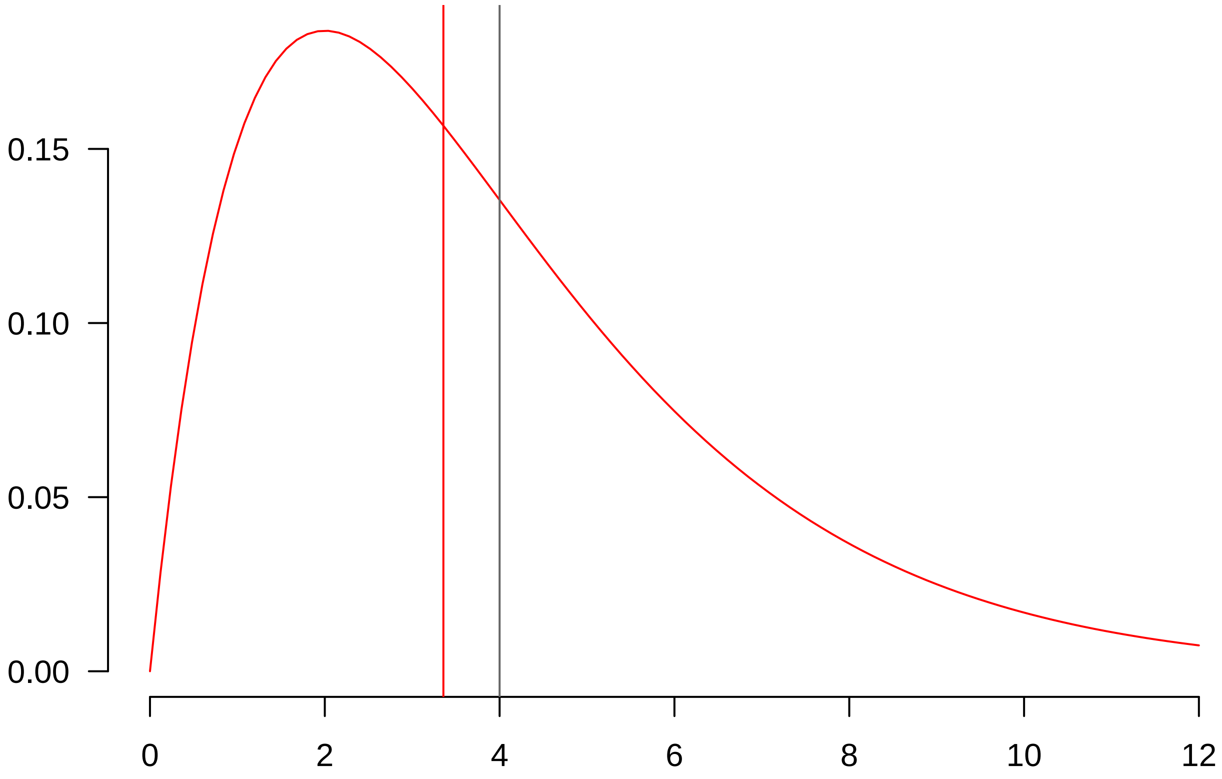
$$\frac{\sqrt{n}(\bar{X} - \mathbb{E}[X])}{s} \approx N(0,1).$$

- This gives me a 95% confidence interval like

$$\bar{X} \pm 2 * s/\sqrt{n}$$

- But I don't want to estimate the mean, I want to estimate the median.

```
par(mar=c(2,3,0,0))
curve(dchisq(x, df=4), from=0, to=12, bty='n', las=1, col=2,ylab='')
abline(v=4, col='grey40') # mean
abline(v=qchisq(.5, 4), col=2) # median
```



Now what

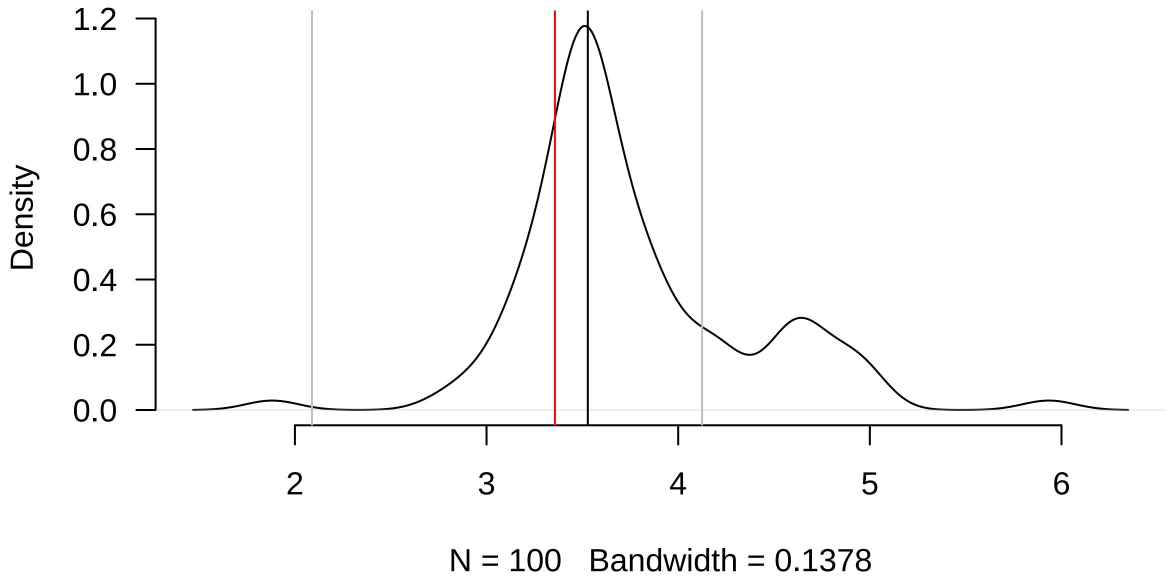
- I give you a sample of size 50, you give me the sample median.
- How do you get a CI?
- You can use the bootstrap!

```
set.seed(2017-02-18)
x = rchisq(n, 4)
(med = median(x))
```

```
## [1] 3.528464
```

```
B = 100
alpha = 0.05
bootMed <- function(x) median(sample(x, length(x), replace=TRUE))
bootDist = replicate(B, bootMed(x))
bootCI = 2* med - quantile(bootDist, probs = c(1-alpha/2, alpha/2))
plot(density(bootDist), bty='n', las=1, main='')
```

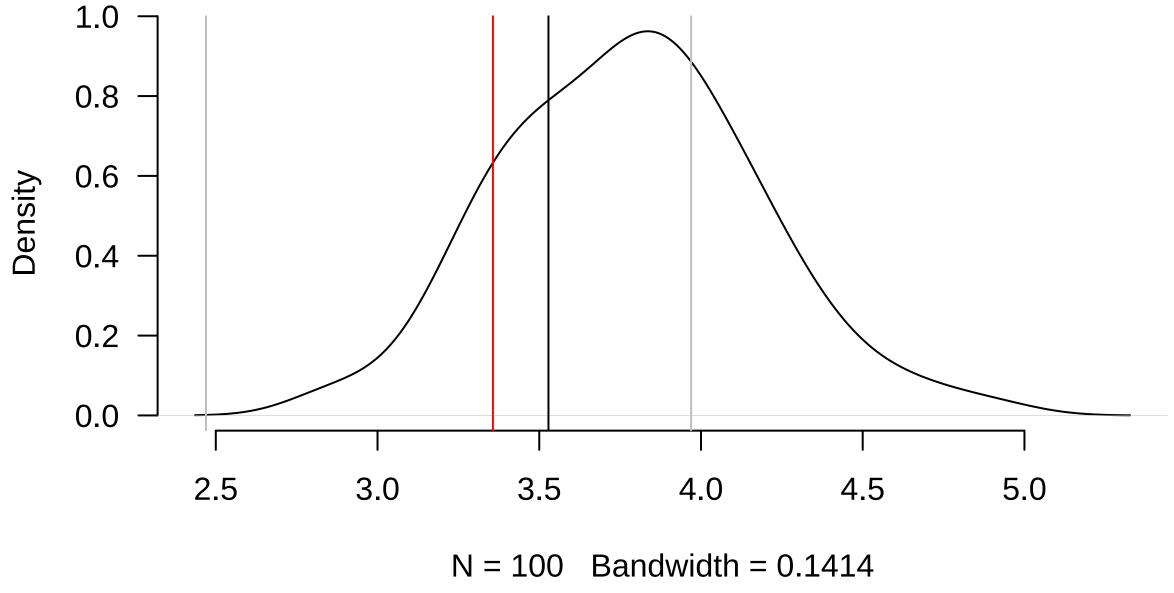
```r
abline(v=bootCI, col='grey')
abline(v=med, col=1)
abline(v=qchisq(.5, 4), col=2)
```



An alternative

- In that bootstrap, I didn't use any information about the data-generating process.
- What if I told you that the data came from a χ^2, but I didn't tell you the degrees of freedom?
- You could try a "parametric" bootstrap:

```r
xbar = mean(x)
s = sd(x)
ParaBootSamp <- function(B, xbar, s){
  means = rnorm(B, mean=xbar, sd=s/sqrt(n))
  meds = qchisq(.5, means)
  return(meds)
}
ParaBootDist = ParaBootSamp(B, xbar, s)
ParaBootCI = 2* med - quantile(ParaBootDist, probs = c(1-alpha/2, alpha/2))
plot(density(ParaBootDist), bty='n', las=1, main='')
abline(v=ParaBootCI, col='grey')
abline(v=med, col=1)
abline(v=qchisq(.5, 4), col=2)
```



In truth

- Let's compare these intervals
- The nonparametric bootstrap (first one) had a width of

```
bootCI[2] - bootCI[1]
```

```
##     2.5%
## 2.035708
```

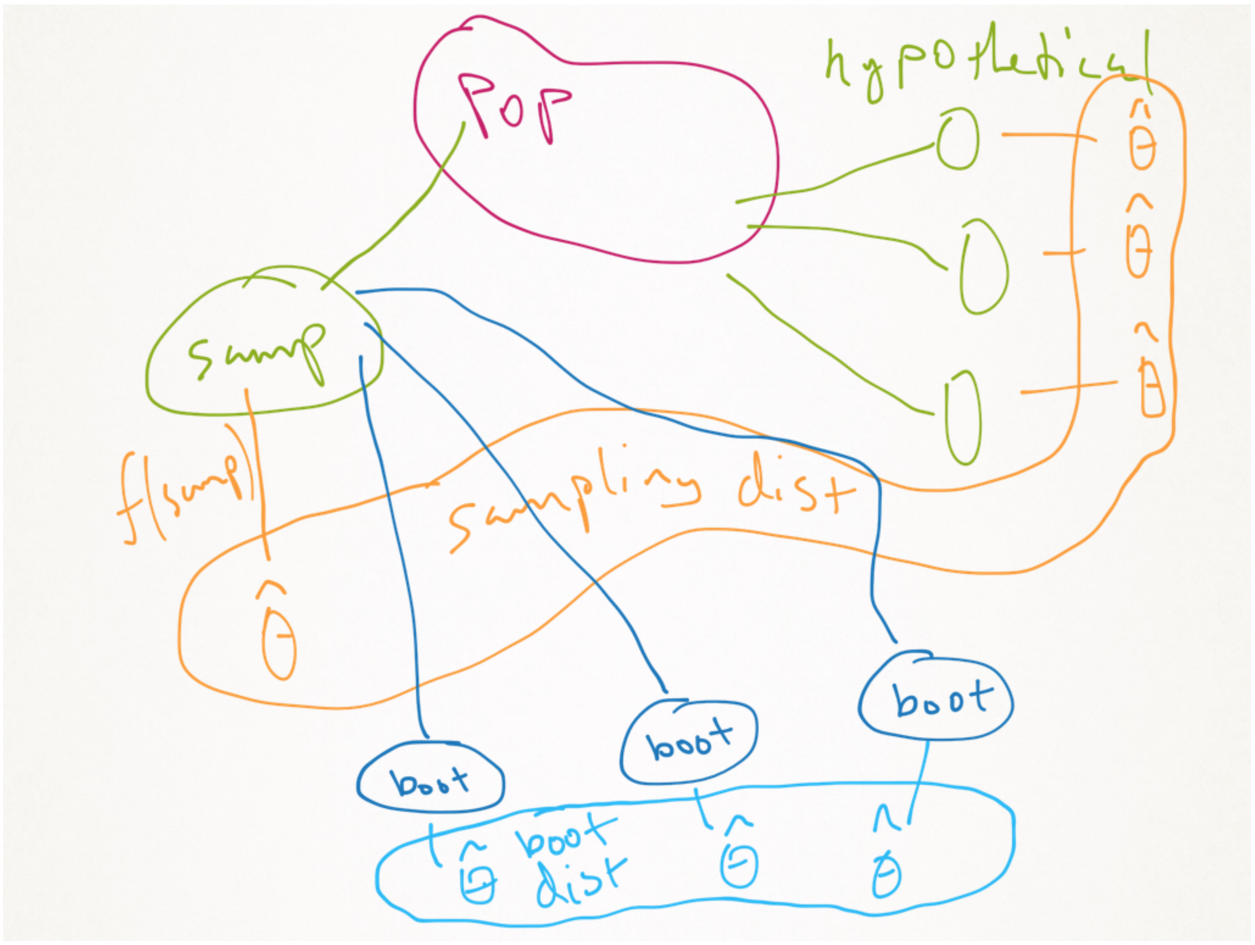
- The parametric bootstrap (second one) had a width of

```
ParaBootCI[2] - ParaBootCI[1]
```

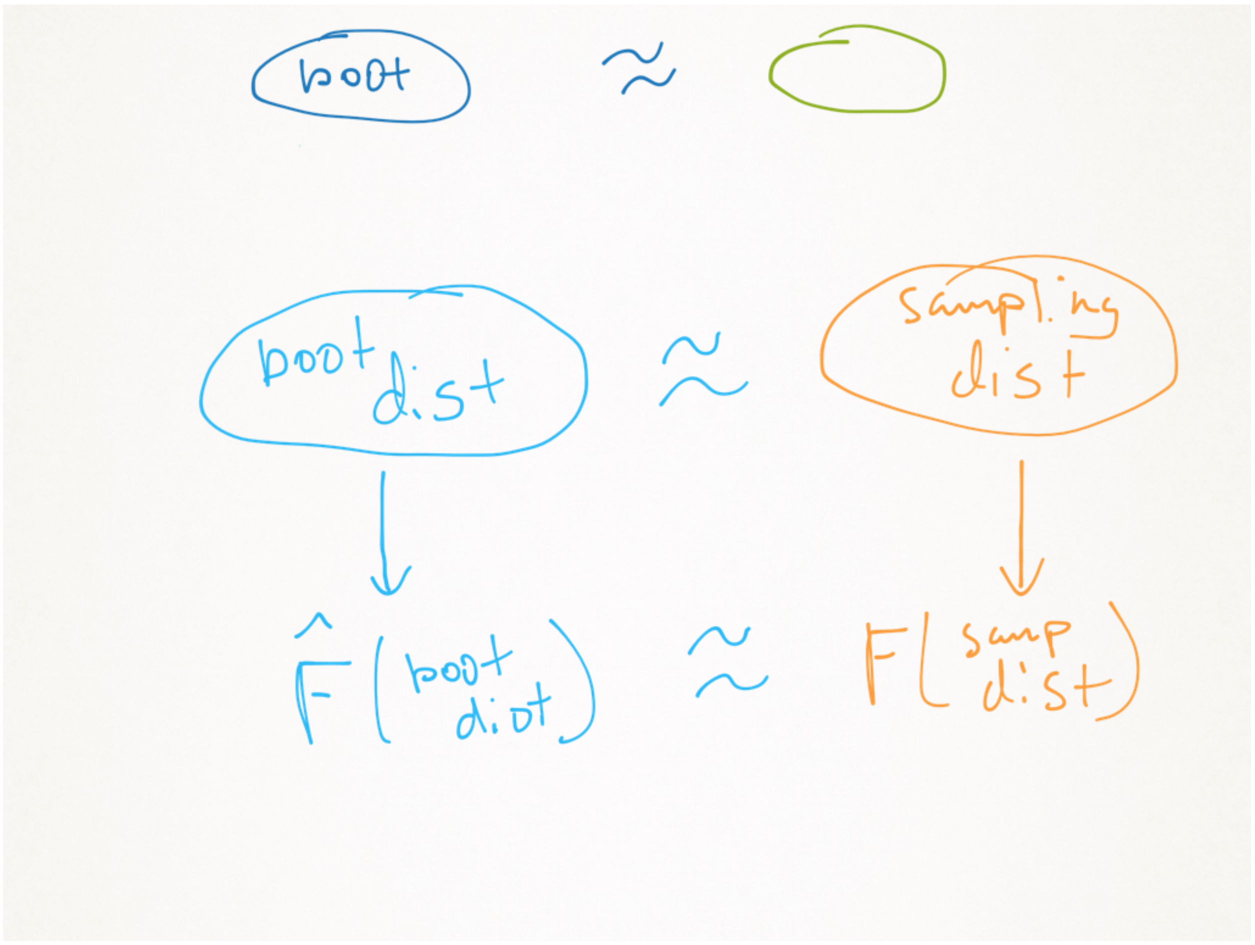
```
##     2.5%
## 1.499788
```

- Using theory, we could find the exact CI. In this case, it has a width of 1.76.

Bootstrap diagram



Bootstrap intuition



Bootstrap error sources

(From the bottom up on the last slide)

1. Simulation error: using only B samples to estimate F with $\hat{F}$.
2. Statistical error: our data depended on a sample from the population. We don't have the whole population so we make an error by using a sample (Note: this part is what **always** happens with data, and what the science of statistics analyzes.)
3. Specification error: If we use the model based bootstrap, and our model is wrong, then we think we are badly overconfident in our assessment of error.

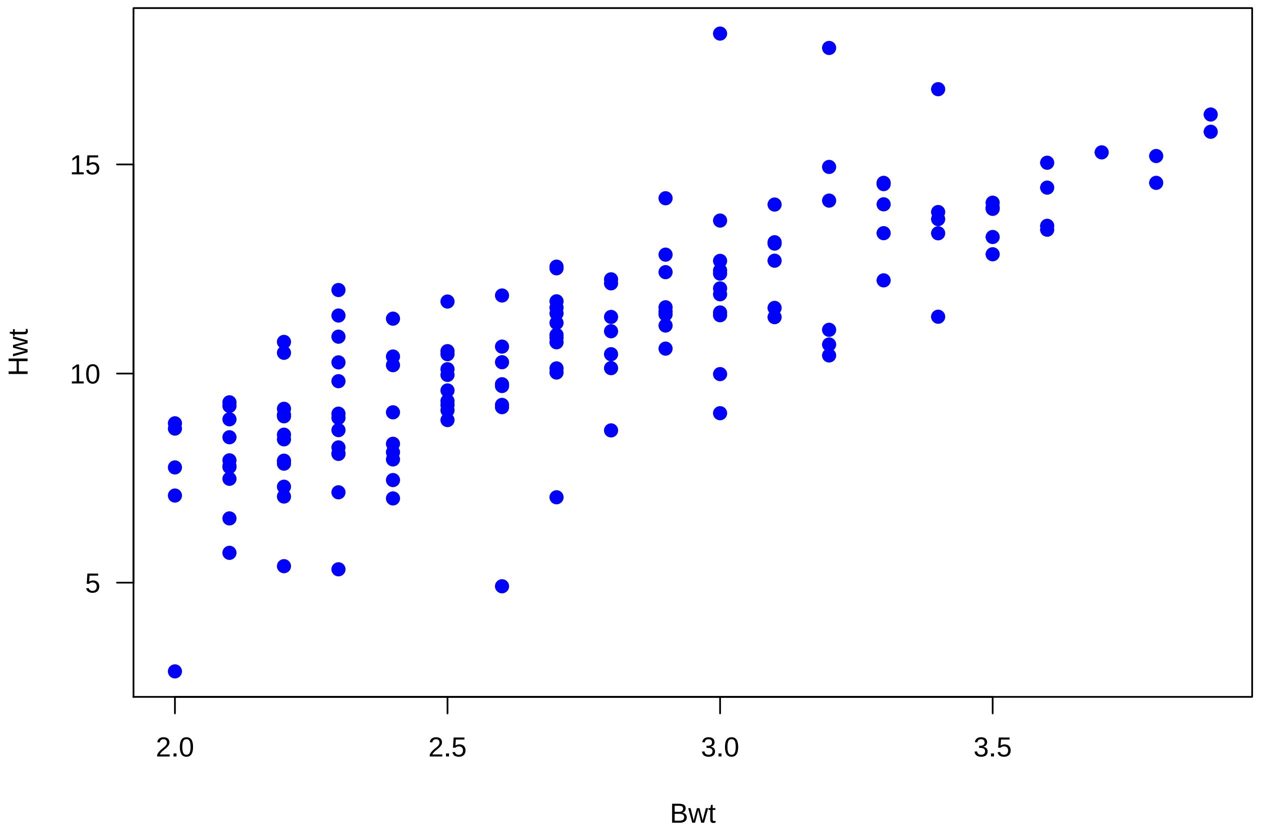
Recap

- There are essentially 2 types of bootstrap
  1. Parametric
  2. Nonparametric
- If you **really** believe your model, use the first
- If not, use the second

- Both are valid

Example 2

```
library(MASS)
plot(Hwt~Bwt, data=fatcats, pch=19, col=4, las=1)
```



A model

```
cats.lm <- lm(Hwt ~ 0+Bwt,data=fatcats)
summary(cats.lm)
```

```
##
## Call:
## lm(formula = Hwt ~ 0 + Bwt, data = fatcats)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.3796 -0.8172  0.0382  0.7717  6.2517
##
## Coefficients:
##     Estimate Std. Error t value Pr(>|t|)
## Bwt  3.95902    0.04689   84.43   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

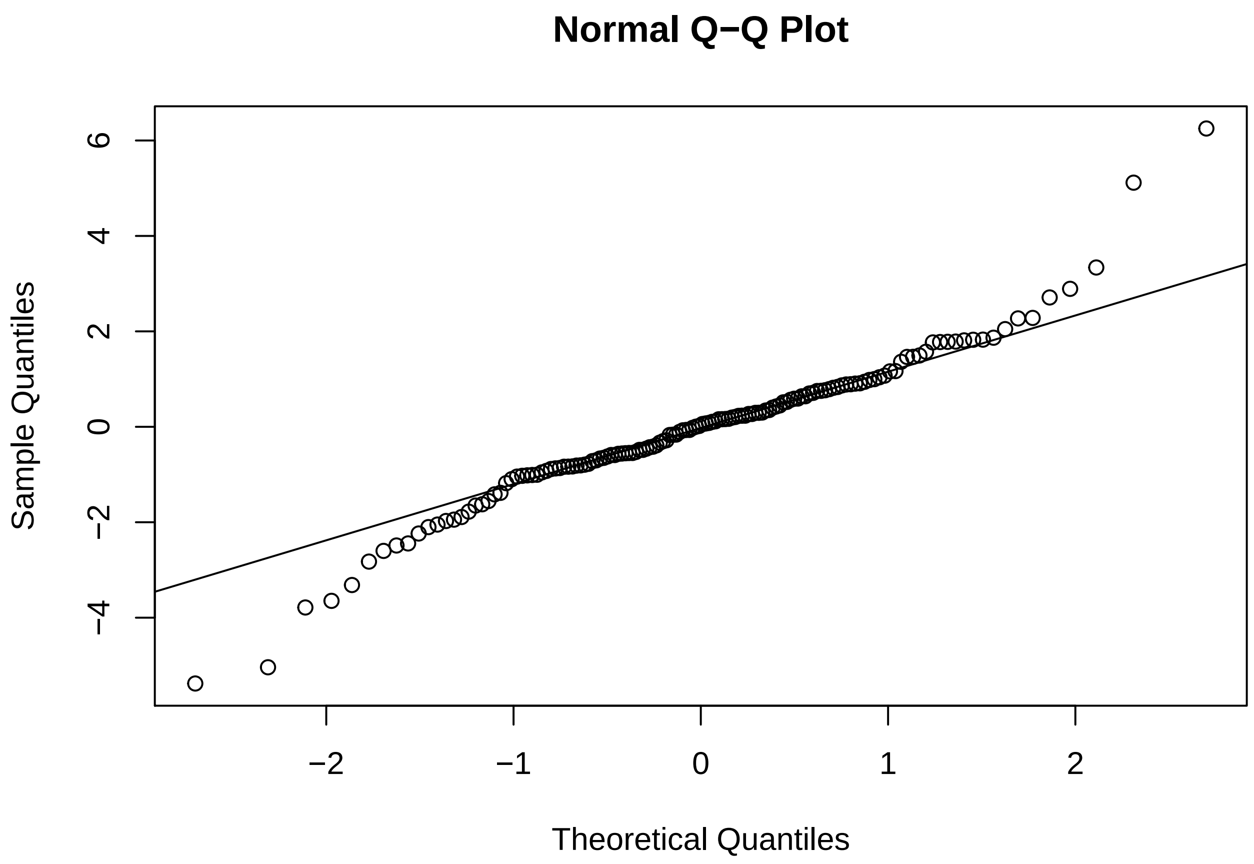
```
##
## Residual standard error: 1.557 on 143 degrees of freedom
## Multiple R-squared:  0.9803, Adjusted R-squared:  0.9802
## F-statistic:  7128 on 1 and 143 DF,  p-value: < 2.2e-16
```

```
confint(cats.lm)
```

```
##        2.5 %   97.5 %
## Bwt 3.866331 4.051717
```

I think that that CI is wrong...

```
qqnorm(residuals(cats.lm))
qqline(residuals(cats.lm))
```



Model-based bootstrap

```
catsResids <- function(){
  resids = residuals(cats.lm)
  newResids = sample(resids, length(resids),
    replace=TRUE) # resample the residuals from the original model
  newCats = data.frame(Bwt = fatcats$Bwt,
    Hwt=fitted(cats.lm) + newResids) # create a new dataframe
                                     # with the original x's but new y's
```

```r
  return(newCats)
}
fitCats <- function(newCats) coef(lm(Hwt~0+Bwt, data=newCats)) # get the coef from OLS
fitCats(fatcats) # test the above on original data, should give same coef
```

```
##      Bwt
## 3.959024
```

Model-based bootstrap

```r
catsBoot <- function(B, alpha, func, bootname){
  coefs = replicate(B, fitCats(func())) # the bootstrap distribution hat(F)
  ci = 2*coef(cats.lm) - quantile(coefs, probs = c(1-alpha/2, alpha/2)) # ci
  cis = rbind(confint(cats.lm), ci) # The rest just for comparison
  rownames(cis) = c('original',bootname)
  return(cis)
}
cisPara = catsBoot(1000,.05, catsResids, 'MBB') # do it!
```

Nonparametric bootstrap

```r
resampData <- function(){
  sampled.rows = sample(1:nrow(fatcats), size=nrow(fatcats),
    replace=TRUE) # resample original data
  new.cats = fatcats[sampled.rows,]
}
cisNonPara = catsBoot(1000, .05,
              resampData, 'NonParaB') # use the prev func to
                                      # bootstrap on resampled data
cisPara
```

```
##             2.5 %   97.5 %
## original 3.866331 4.051717
## MBB      3.885491 4.071610
```

```r
cisNonPara
```

```
##             2.5 %   97.5 %
## original 3.866331 4.051717
## NonParaB 3.872128 4.047525
```

Bootstrapping with nonparametric regression

- This is a bit harder
- The reason is that we use CV to choose the bandwidth
- So we have to repeat that step in the bootstrapping
- That is:
    1. Input data
    2. Use CV to choose a smoothing parameter

3. Use the chosen parameter to estimate the smooth function
4. Resample the data
5. Using this new data, repeat 2 and 3