STATS 330

Handout 13
Bootstrapping—replacing theory with computer power

STATS 330

Department of Statistics, University of Auckland

Introduction

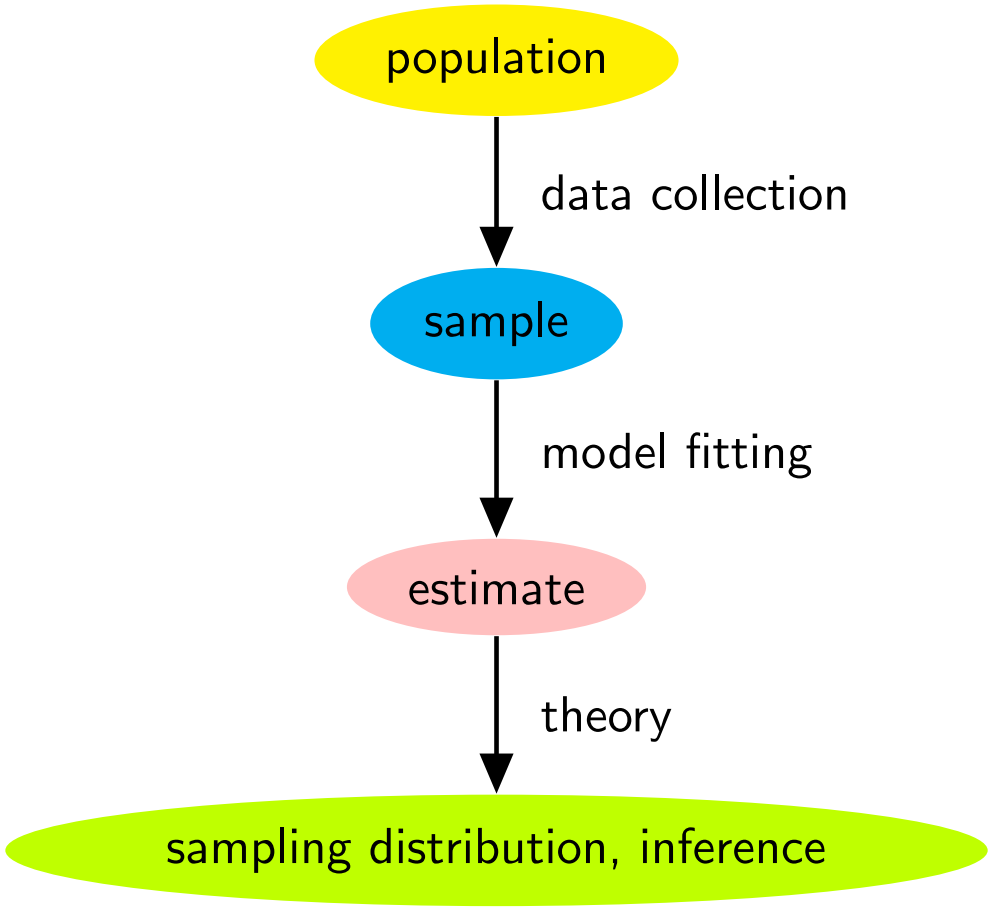
在之前的课程中，我们一直在使用模拟来学习各种统计量的抽样分布。我们假设我们知道真实的关系和真实的总体参数。
然而，在现实中，我们很少知道真实的关系，所以我们对收集到的数据进行拟合。
为了了解我们拟合的模型的可靠性并从中做出推断，我们进行其他计算，例如获得标准误差、置信区间和模型检查等。

In the last few classes we have been looking at the value of simulation for learning about the sampling distribution of various statistics. In all cases, we pretended that we knew the 'true' relationship and the 'true' population parameters.

Of course, in the real world we seldom know the true relationship. So, instead we fit a model. That is, we collect some data and fit a parsimonious model to those data.

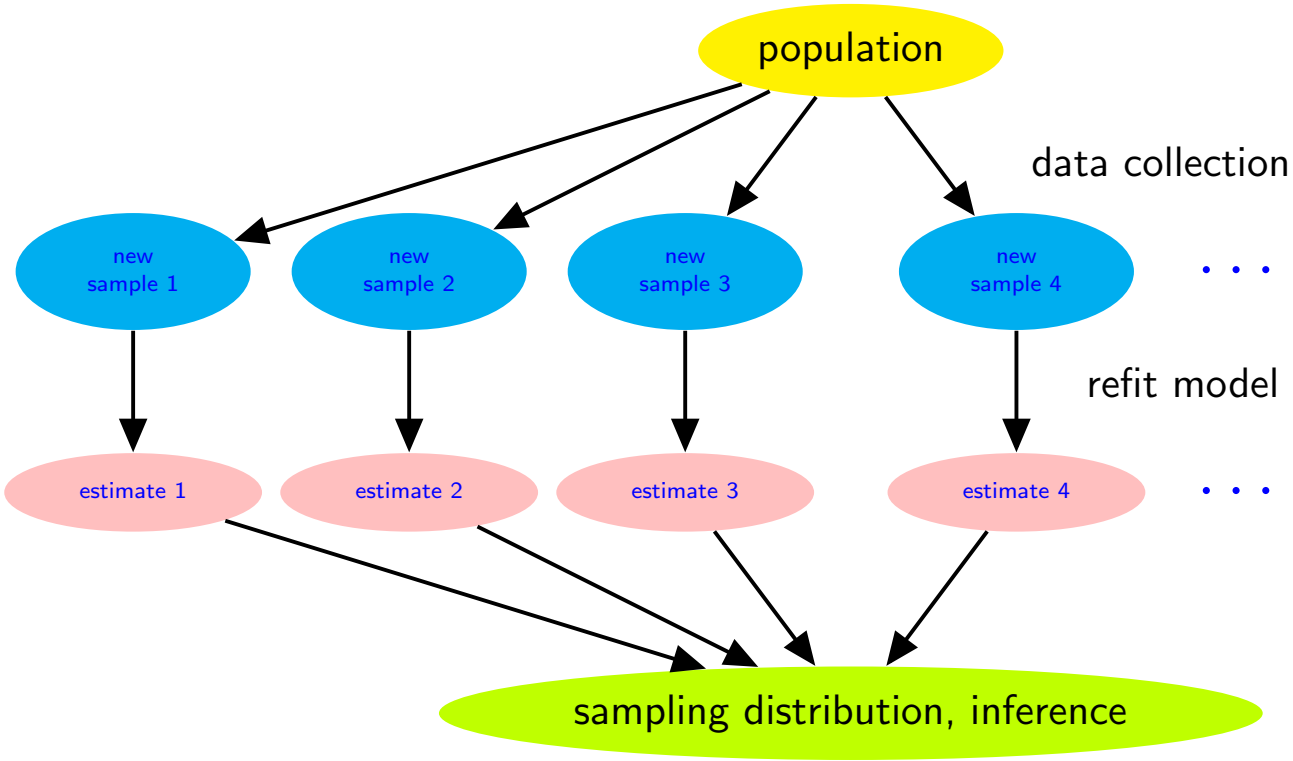
To help us understand the reliability of our fitted model, and to make inference from it, we then do other calculations, such as obtaining standard errors, confidence intervals, model checking, etc.

Traditional Statistical Inference



- Each arrow is an opportunity for things to go wrong. Can you identify what might go wrong at each step?

Hypothetical Ideal Situation for Inference



- Ideal but utterly impractical.

Introduction

So far, the procedures we've used to get standard errors, calculate CIs, do model checking (etc.) have relied on formulae obtained from statistical theory.

We've already seen that these formulae may not be valid in some situations (e.g., GLMs when the data are sparse). For more complex models the theory may not even exist! What should we do then???

到目前为止，我们使用的获取标准误差、计算置信区间和进行模型检查等程序都依赖于从统计理论获得的公式。
我们已经看到，这些公式在某些情况下可能不适用，例如在数据稀疏的GLMs中。

对于更复杂的模型，理论甚至可能不存在。我们应该怎么办？

Introduction

So far, the procedures we've used to get standard errors, calculate CIs, do model checking (etc.) have relied on formulae obtained from statistical theory.

We've already seen that these formulae may not be valid in some situations (e.g., GLMs when the data are sparse). For more complex models the theory may not even exist! What should we do then???

- Wouldn't it be great if we could use some kind of simulation instead.

Introduction

So far, the procedures we've used to get standard errors, calculate CIs, do model checking (etc.) have relied on formulae obtained from statistical theory.

We've already seen that these formulae may not be valid in some situations (e.g., GLMs when the data are sparse). For more complex models the theory may not even exist! What should we do then???

- Wouldn't it be great if we could use some kind of simulation instead.
  - We can! **Bootstrapping**.

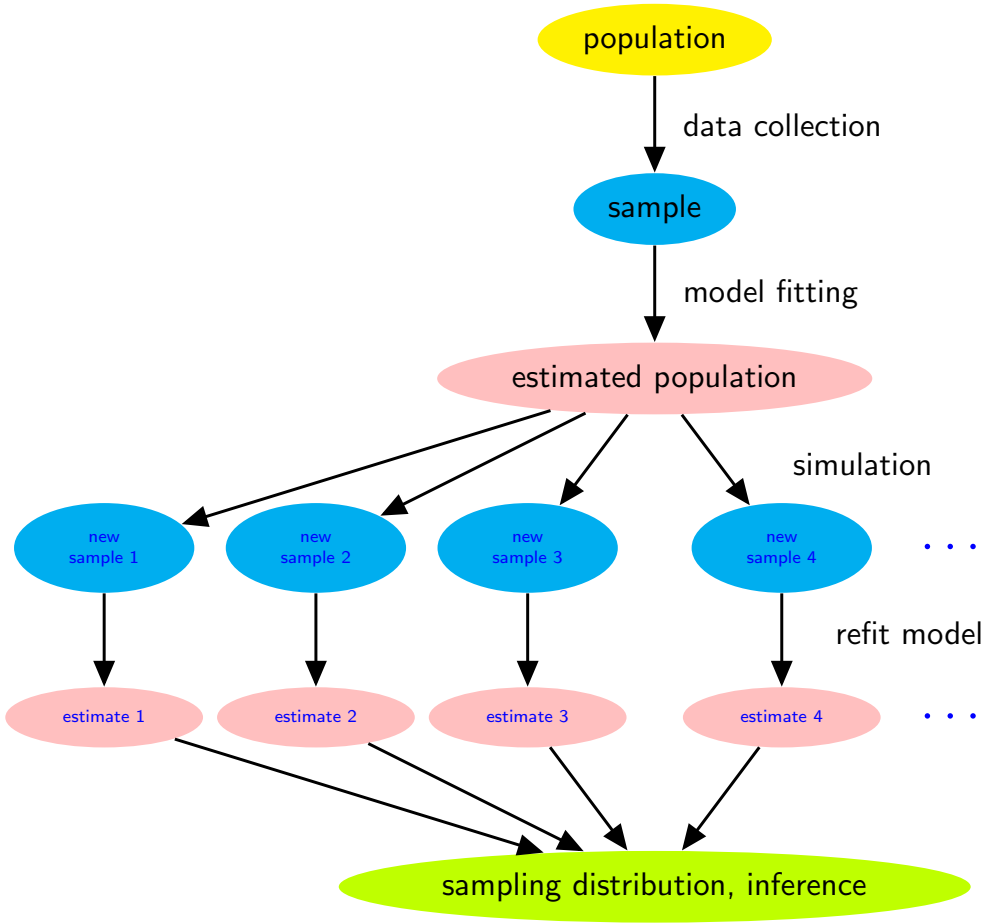
Introduction

So far, the procedures we've used to get standard errors, calculate CIs, do model checking (etc.) have relied on formulae obtained from statistical theory.

We've already seen that these formulae may not be valid in some situations (e.g., GLMs when the data are sparse). For more complex models the theory may not even exist! What should we do then???

- Wouldn't it be great if we could use some kind of simulation instead.
  - We can! **Bootstrapping**.
  - **Parametric bootstrapping** is simply simulation based on the fitted model.

Parametric Bootstrap Inference



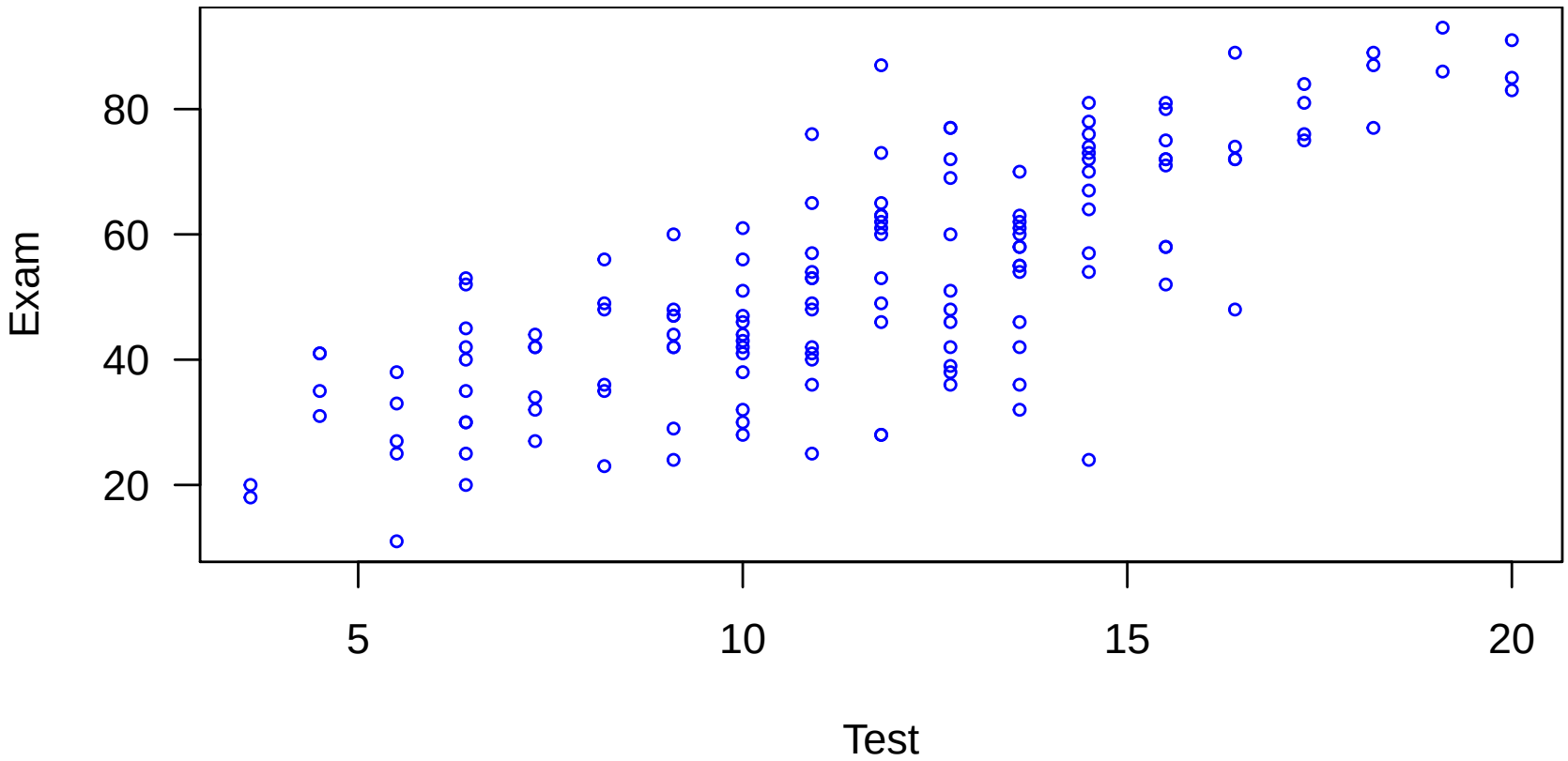
- No longer rely on theory to produce a sampling distribution.

The Parametric Bootstrap

Simple Linear Regression Example

For this example, we're going to "regress" back to STATS20x, and use the class data.

```
library("s20x")
Stats20x.df = read.table("../data/STATS20x.txt", header = T)
plot(Exam ~ Test, data = Stats20x.df, cex=0.6, col="blue")
```



Simple Linear Regression Parametric Bootstrap

The fitted intercept, $\hat{\beta}_0$, and slope, $\hat{\beta}_1$, are:

```
lm.fit = lm(Exam ~ Test, data = Stats20x.df)
betahat = coef(lm.fit)
betahat
```

```
## (Intercept)        Test
##       9.084       3.786
```

and the estimated residual standard error, $\hat{\sigma}$, is

```
sigmahat <- summary(lm.fit)$sigma
sigmahat
```

```
## [1] 12.05
```

We're now going to do a simulation of a simple linear model, using the same explanatory variables `x`, and with our "true" values of the model parameters set to $\beta_0 = 9.0845$, $\beta_1 = 3.7859$, and $\sigma = 12.0477$.

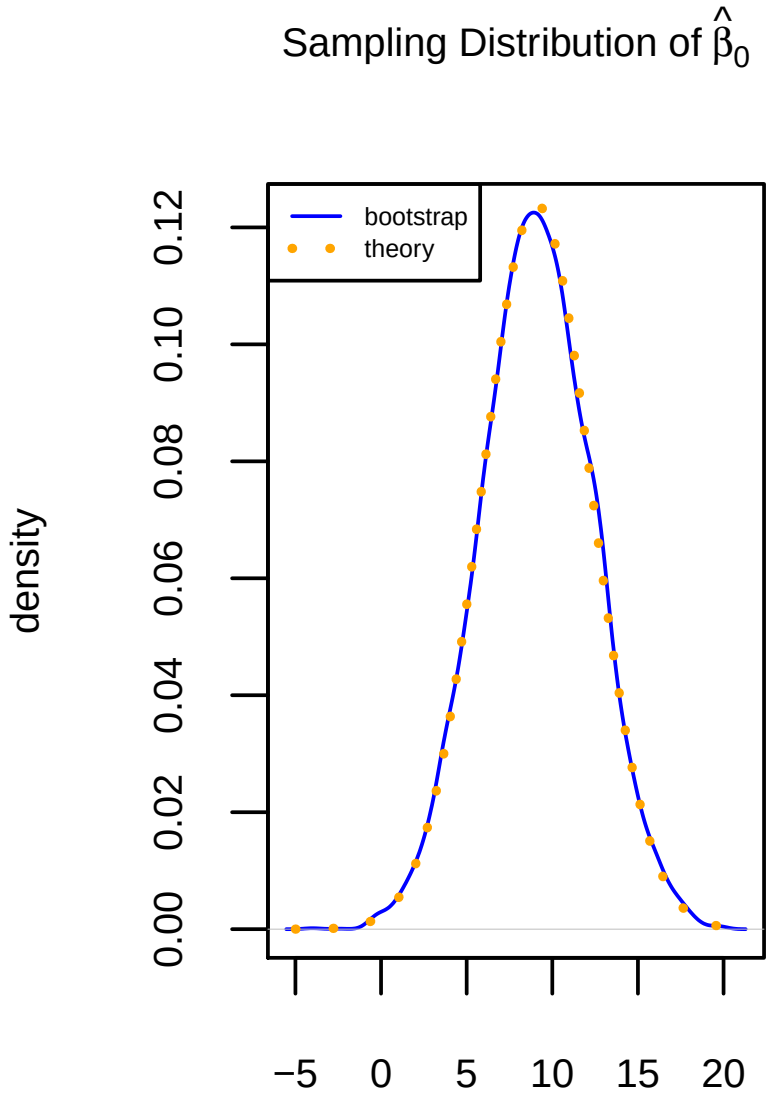
Simple Linear Regression Parametric Bootstrap

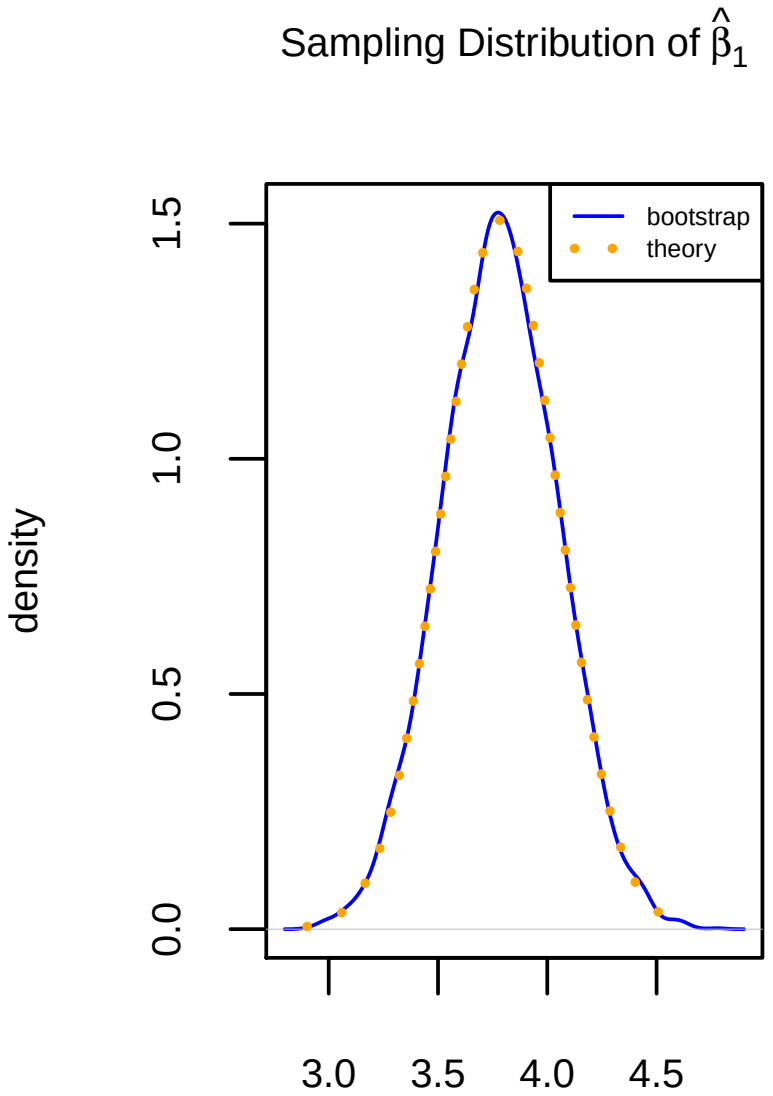
Simulation Code

```
n = nrow(Stats20x.df)  ## Number of observations
## Setting the model's parameters.
b0 <- betahat[1]; b1 <- betahat[2]; sigma <- sigmahat
## Calculating expected response values.
x <- Stats20x.df$Test
means <- b0 + b1*x
## Setting the number of data sets and models to simulate.
n.sims <- 10000
## Creating vectors in which to store estimates.
est.b0 <- est.b1 <- numeric(n.sims)
for (i in 1:n.sims) {  ## The for loop.
  ## Simulating the responses.
  y <- rnorm(n, means, sigma)
  ## Fitting the model.
  fit <- lm(y ~ x)
  ## Extracting the model parameters.
  est.b0[i] <- coef(fit)[1]
  est.b1[i] <- coef(fit)[2]
}
```

Simple Linear Regression Parametric Bootstrap

Bootstrap Generated Sampling Distributions





Simple Linear Regression Parametric Bootstrap

Bootstrap estimate of standard error

Let's have a look at the bootstrap std errors of our simulated regression coefficients:

```
sd(est.b0)
## [1] 3.225
sd(est.b1)
## [1] 0.2651
```

How does this compare to the estimated std errors from `lm.fit`?

```
coef(summary(lm.fit))
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept)    9.084     3.2204   2.821 5.466e-03
## Test           3.786     0.2647  14.301 1.985e-29
```

Pretty darn close!

Simple Linear Regression Parametric Bootstrap

Bootstrap estimate of standard error

In fact, it can be shown that the bootstrap standard error converges to the estimated standard error as the number of bootstrap simulations becomes large.

- We've just seen that we can obtain the (estimated) standard errors of our regression coefficients without knowing any formulae! Instead, we just simulate from the fitted model.

Simple Linear Regression Parametric Bootstrap

Parametric bootstrap confidence intervals

Now, let's take a look at the quantile 95% confidence intervals of the bootstrap values of our regression coefficients.

```
quantile(est.b0, prob=c(0.025, 0.975))
##  2.5% 97.5%
##  2.79 15.48
quantile(est.b1, prob=c(0.025, 0.975))
##  2.5% 97.5%
## 3.269 4.307
```

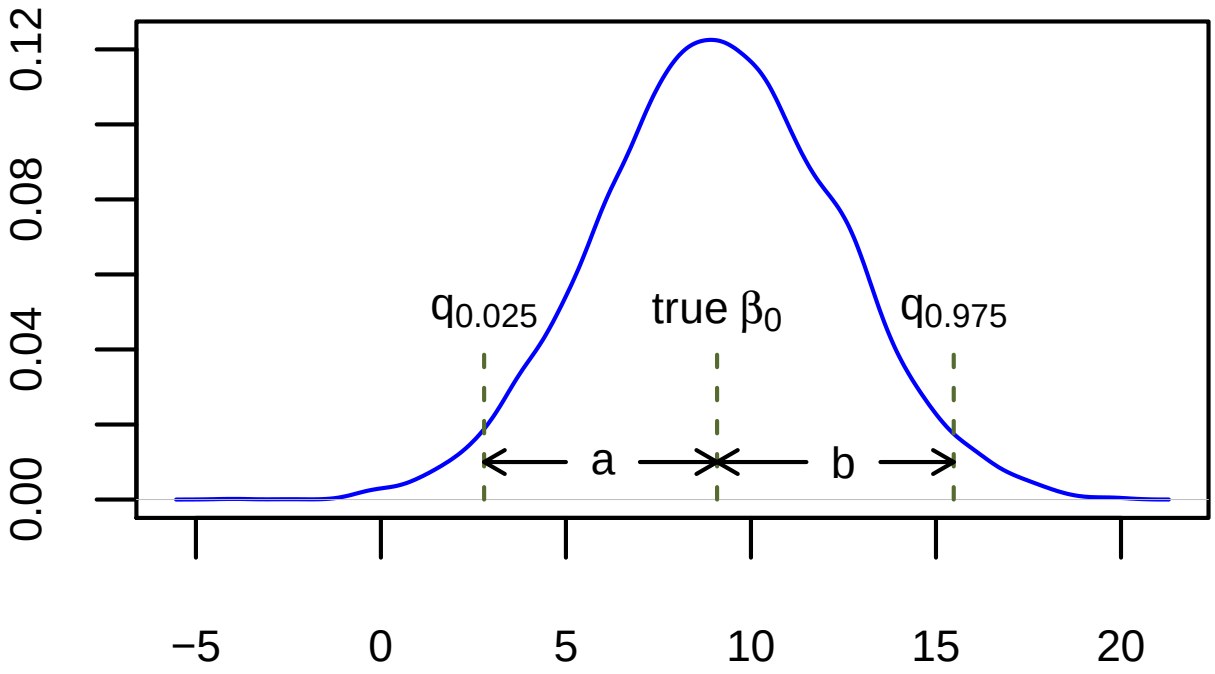
How does this compare to the confidence intervals from `lm.fit`...

```
confint(lm.fit)
##                2.5 % 97.5 %
## (Intercept) 2.719 15.450
## Test        3.263  4.309
```

Wow! Once again, we see that we don't have to know any formulae for calculating confidence intervals. Our computer can calculate confidence intervals using brute computational force.

Invert the confidence intervals? I

The confidence intervals we just generated should really be "inverted".



- Our generated sampling distribution indicates that with 95% probability an estimate $\hat{\beta}_0$ will be between $\beta_0 - a$ and $\beta_0 + b$.

Invert the confidence intervals? II

Starting from the estimated value, this means that with 95% probability the true value β_0 will be between $\hat{\beta}_0 - b$ and $\hat{\beta}_0 + a$.

- If our bootstrap generated sampling distribution is symmetric (more or less), then it doesn't make much difference whether we invert the interval or not.
- However if the sampling distribution is skewed, it is important that we invert the interval.
- The key is to remember that when we use parametric bootstrapping to produce sampling distributions, we treat the estimated coefficients as if they were the true values. But when we create the confidence intervals we go back to treating them as estimated values.

Invert the confidence intervals? III

To invert our parametric bootstrap interval for β_0:

```
a = b0 - quantile(est.b0, prob=0.025)
b = quantile(est.b0, prob=0.975) - b0
c(b0-b, b0+a)
```

```
## (Intercept) (Intercept)
##       2.689      15.379
```

An equivalent way of doing this calculation is:

```
c(2*b0-quantile(est.b0, prob=0.975), 2*b0-quantile(est.b0, prob=0.025))
```

```
## (Intercept) (Intercept)
##       2.689      15.379
```

To invert our parametric bootstrap interval for β_1:

```
c(2*b1-quantile(est.b1, prob=0.975), 2*b1-quantile(est.b1, prob=0.025))
```

```
##  Test  Test
## 3.265 4.303
```

The Principle of Bootstrapping

There is a large literature of scientific publications that establishes the properties of bootstrapping[1].

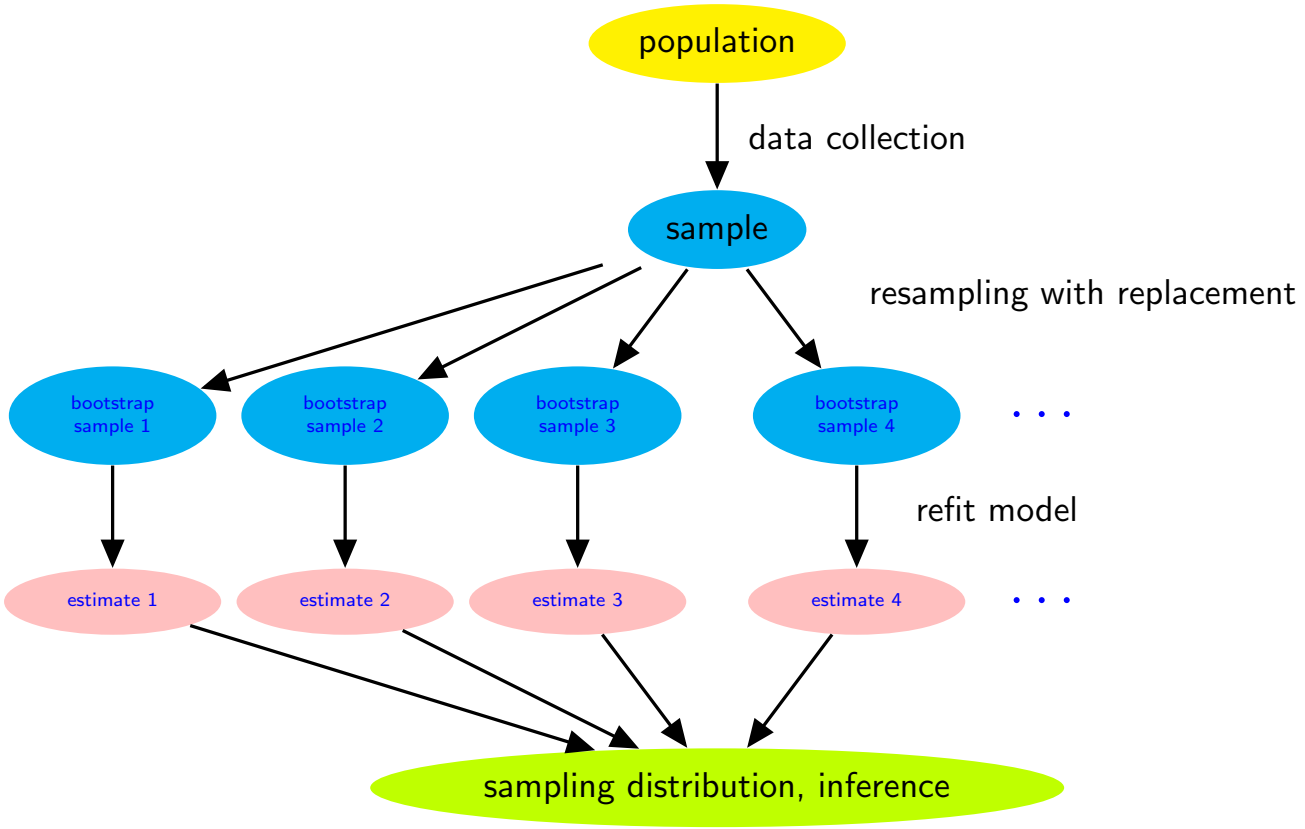
The underlying principle of bootstrapping is that we emulate the "experiment" on the computer. That is, we emulate the observation of data, and the fitting of a model to those data. This is done many many times to generate sampling distributions for the statistics we are trying to estimate.

In the above, we used parametric bootstrapping to emulate the "experiment". When the data are independent, there is another way we can do this emulation—the **non-parametric bootstrap**.

The non-parametric bootstrap simply generates new "data" *by resampling with replacement* from the actual data.

[1] A little bit of this is seen in STATS 730

Non-Parametric Bootstrap Inference



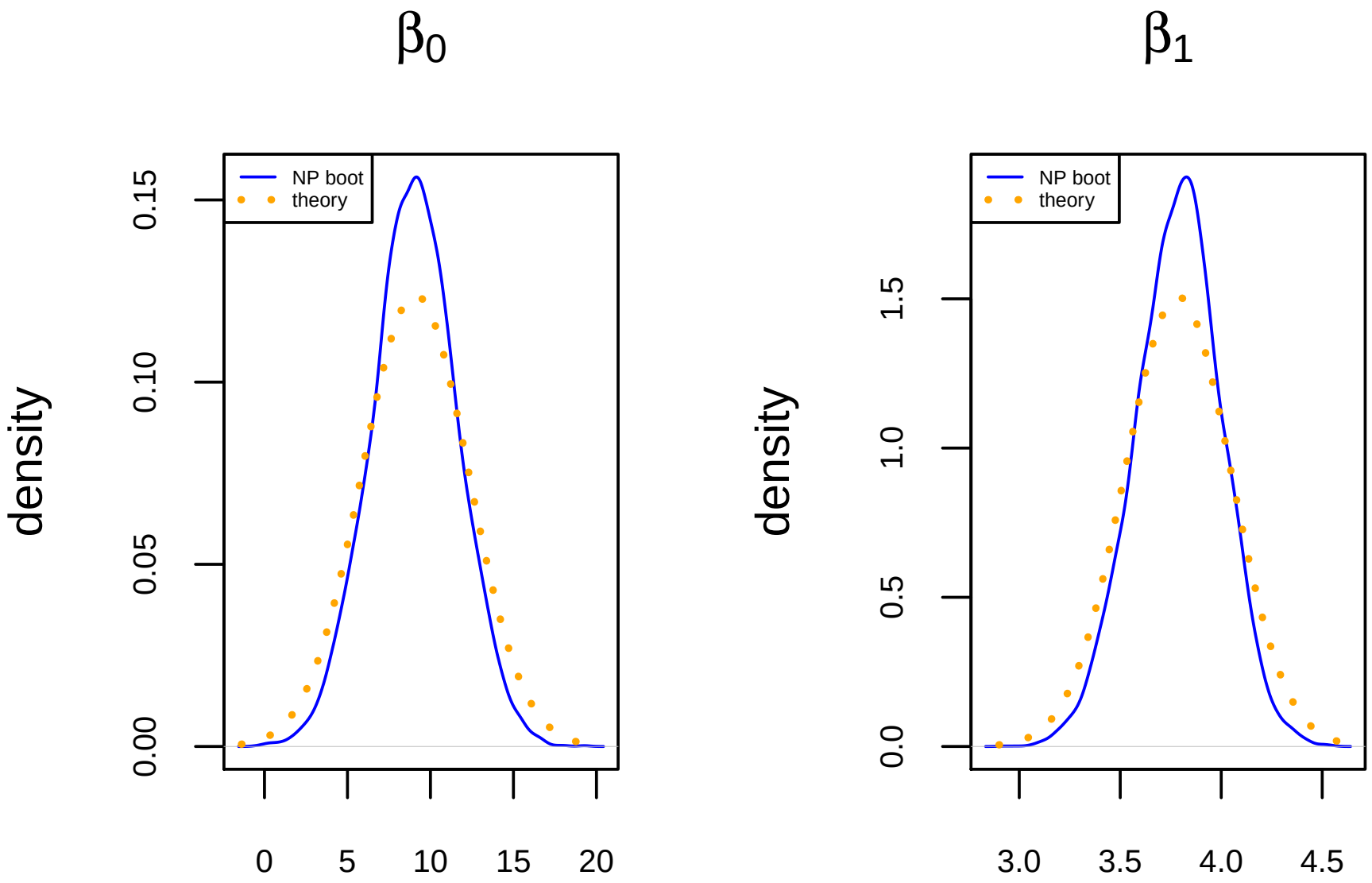
- Don't need to assume that the estimated model is the true relationship.

Simple linear regression non-parametric bootstrap

Simulation code

```
## Number of observations
n = nrow(Stats20x.df)
## Setting the number of data sets and models to simulate.
n.sims <- 10000
## Creating vectors in which to store estimates.
np.est.b0 <- np.est.b1 <- numeric(n.sims)
for (i in 1:n.sims) {  ## The for loop.
    ## Resampling rows of the dataframe
    samp = sample(1:n, replace=TRUE)
    Boot.df <- Stats20x.df[samp, ]
    ## Fitting the model.
    fit <- lm(Exam ~ Test, data=Boot.df)
    ## Extracting the model parameters.
    np.est.b0[i] <- coef(fit)[1]
    np.est.b1[i] <- coef(fit)[2]
}
```

Simple linear regression non-parametric bootstrap



Simple linear regression non-parametric bootstrap

Bootstrap estimate of standard error

Let's have a look at the bootstrap std errors of our simulated regression coefficients:

```
sd(np.est.b0)
## [1] 2.582
sd(np.est.b1)
## [1] 0.2142
```

How does this compare to the estimated std errors from the parametric bootstrap:

```
sd(est.b0)
## [1] 3.225
sd(est.b1)
## [1] 0.2651
```

Non-parametric bootstrap std errors are somewhat smaller.

Simple linear regression non-parametric bootstrap

Bootstrap confidence intervals

Now, look at the (inverted) 95% quantile intervals of the non-parametric bootstrap values of our regression coefficients.

```
c(2*b0-quantile(np.est.b0, prob=0.975), 2*b0-quantile(np.est.b0, prob=0.025))
## (Intercept) (Intercept)
##       4.062      14.200
c(2*b1-quantile(np.est.b1, prob=0.975), 2*b1-quantile(np.est.b1, prob=0.025))
##  Test  Test
## 3.376 4.215
```

Compare to the (inverted) parametric bootstrap confidence intervals:

```
c(2*b0-quantile(est.b0, prob=0.975), 2*b0-quantile(est.b0, prob=0.025))
## (Intercept) (Intercept)
##       2.689      15.379
c(2*b1-quantile(est.b1, prob=0.975), 2*b1-quantile(est.b1, prob=0.025))
##  Test  Test
## 3.265 4.303
```

- The non-parametric CIs are somewhat narrower.

Parametric vs non-parametric bootstrap

If the fitted model is good then the parametric and non-parametric bootstraps will give similar results.

In this example, the difference between the parametric and non-parametric bootstraps is due to weaknesses with the fitted linear model. Is the parametric bootstrap better, or the non-parametric bootstrap?

The answer comes down to asking which bootstrap better emulates the experiment.

Parametric vs non-parametric bootstrap

If the fitted model is good then the parametric and non-parametric bootstraps will give similar results.

In this example, the difference between the parametric and non-parametric bootstraps is due to weaknesses with the fitted linear model. Is the parametric bootstrap better, or the non-parametric bootstrap?

The answer comes down to asking which bootstrap better emulates the experiment.

- Generating normally distributed exam scores around the fitted line (parametric bootstrap).

Parametric vs non-parametric bootstrap

If the fitted model is good then the parametric and non-parametric bootstraps will give similar results.

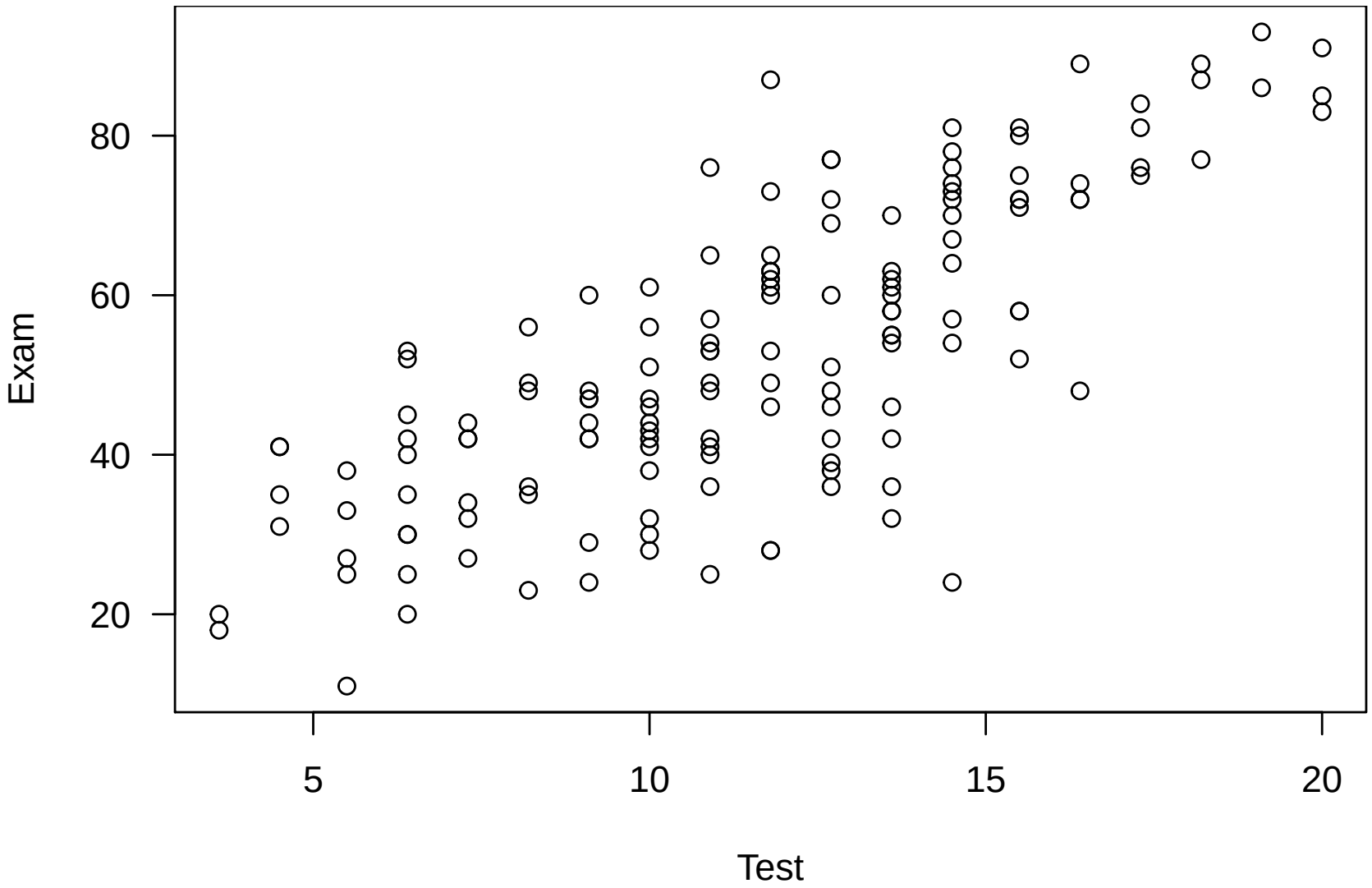
In this example, the difference between the parametric and non-parametric bootstraps is due to weaknesses with the fitted linear model. Is the parametric bootstrap better, or the non-parametric bootstrap?

The answer comes down to asking which bootstrap better emulates the experiment.

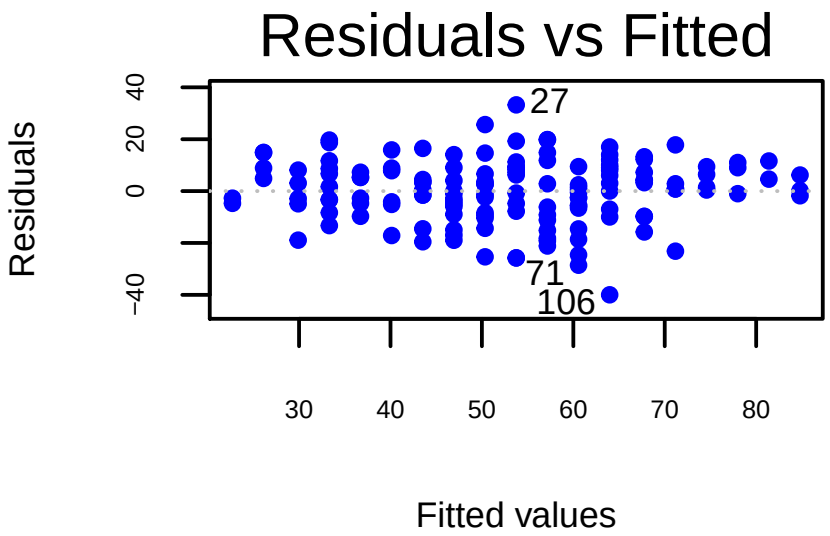
- Generating normally distributed exam scores around the fitted line (parametric bootstrap).
- Sampling from a population consisting of infinite copies of the Stats20x data (non-parametric bootstrap).

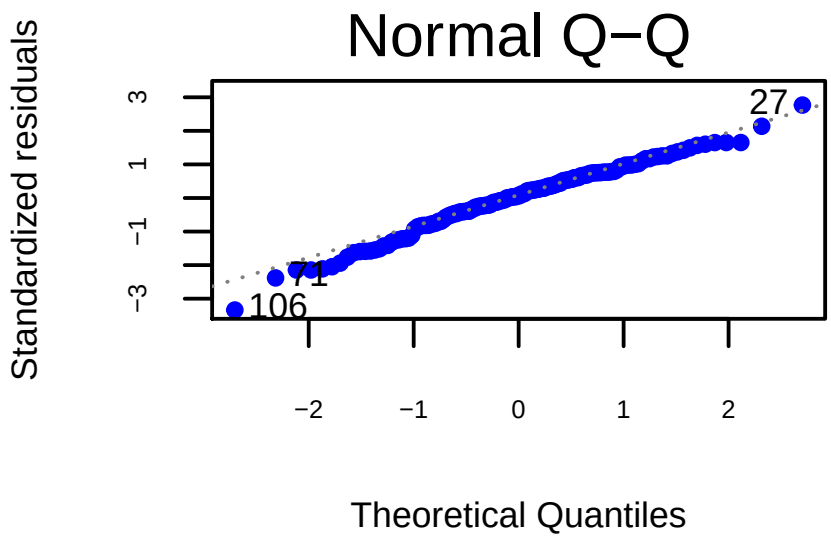
Parametric vs non-parametric bootstrap

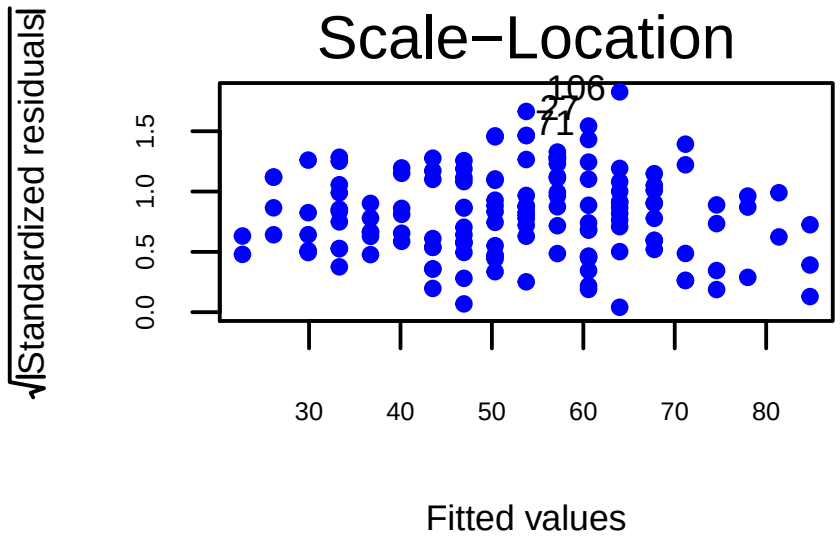
Another look at a scatter plot of the data may help to assess the situation. . .

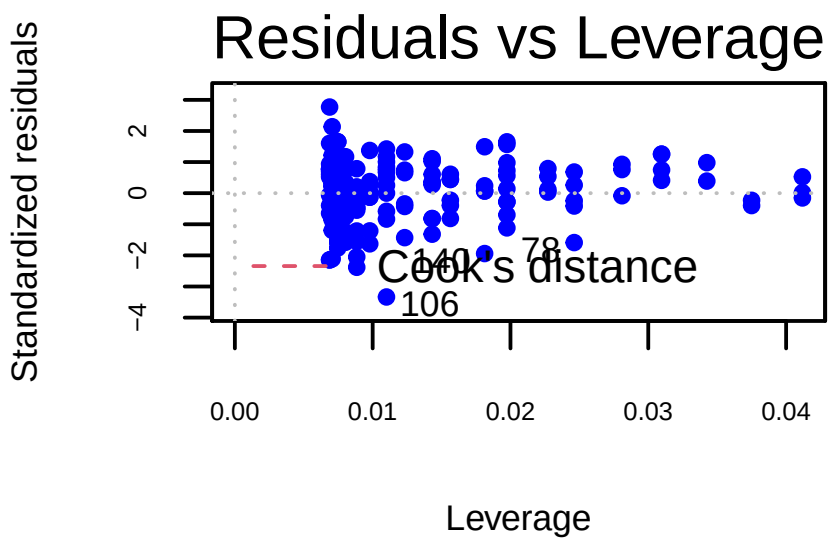


Diagnostic Plots









Summary

Bootstrapping allows the sampling distribution of almost any statistic to be estimated empirically. It is particularly useful in cases where:

- It is difficult or impossible to obtain a sampling distribution based on theory.
- The assumptions required to obtain a theoretical sampling distribution are in doubt.

Some important points:

- The bootstrap is NOT used to get a better estimate of a statistic—the expected value of the bootstrap estimate is equal to the sample estimate.
- The data must be independent and representative of the population.
- Does not compensate for model misspecification.

Bootstrapping examples

The remainder of this handout presents of using boostrapping with real data sets.

Coronary heart disease data revisited

A sample of 100 subjects were tested for presence of significant coronary heart disease (CHD).

The data were grouped prior to analysis, so that subjects of the same age form a group. The data frame contains the following variables:

`age`: The age of subjects in a particular group in years.

`n`: The number of subjects in a particular age group.

`y`: The number of subjects in a particular age group with significant CHD.

`p`: The proportion of subjects in a particular age group with significant CHD (`y`/`n`).

Coronary heart disease data I

The data

We will import these data, model it and plot the predicted values for these proportions.

```
> head(chd.df, 4)
```

```
##   age y n   p
## 1  20 0 1 0.0
## 2  23 0 1 0.0
## 3  24 0 1 0.0
## 4  25 1 2 0.5
```

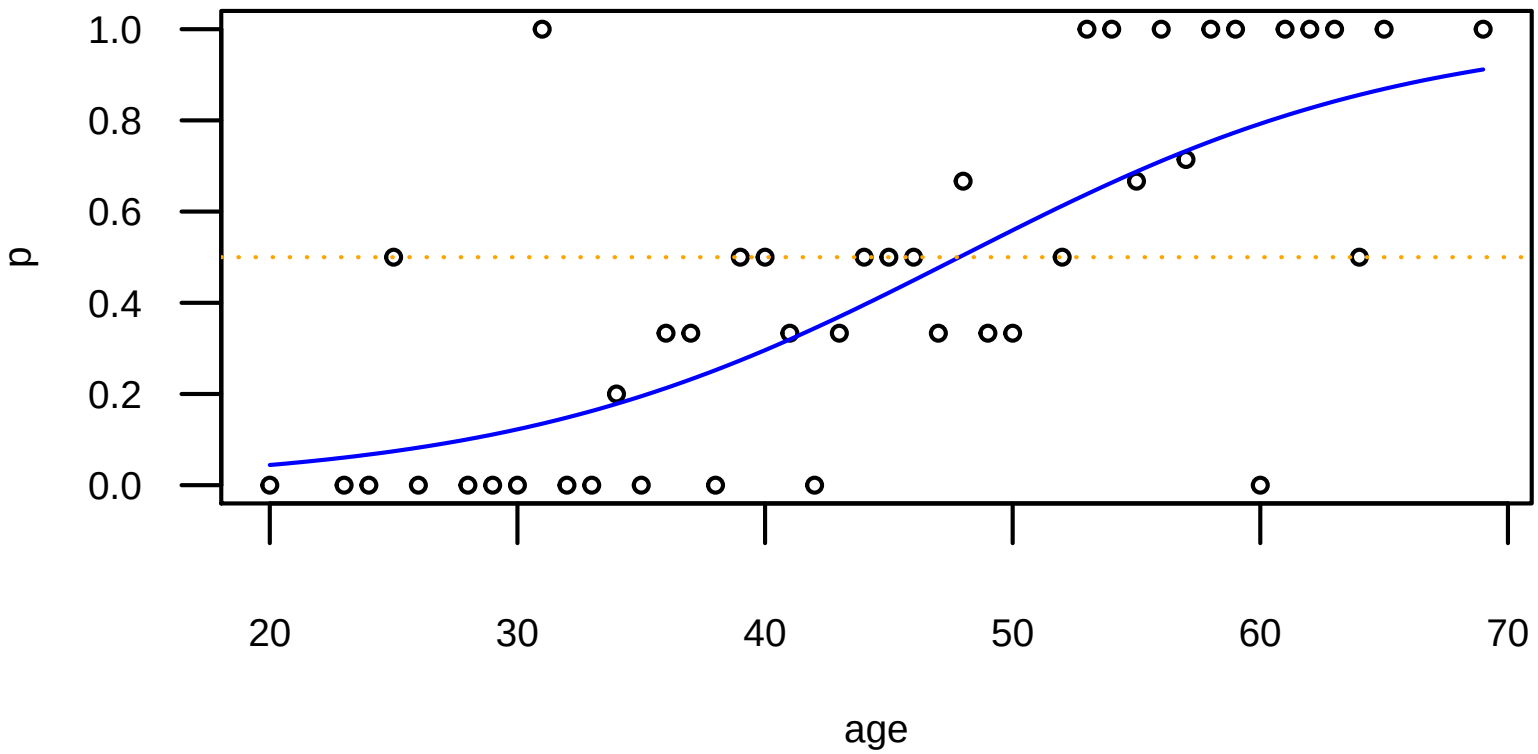
```
> chd.fit = glm(cbind(y, n - y) ~ age, binomial, data=chd.df)
> slimSummary(chd.fit)
```

```
## Call:
## glm(formula = cbind(y, n - y) ~ age, family = binomial, data = chd.df)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -5.278      1.131   -4.67  3.0e-06 ***
## age            0.110      0.024    4.59  4.4e-06 ***
## ---
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 63.958  on 42  degrees of freedom
## Residual deviance: 34.976  on 41  degrees of freedom
```

Coronary heart disease data II

The data

```
plot(p~age, data=chd.df, ylim = 0:1, las = 1, mgp = c(2.2, 1, 0), cex=0.5, cex.axis=0.6, cex.lab=0.6)
Ages = with(chd.df, seq(min(age), max(age), length=200))
preds = predict(chd.fit, newdata=data.frame(age=Ages), type="response")
lines(Ages, preds, col="blue")
abline(h=0.5, lty=3, col="orange")
```



Coronary heart disease data I

The question of interest

Suppose we wish to estimate the age at which there is a 50% chance of being diagnosed as having CHD. This is the same as obtaining the value of age associated with $\mu = p = \frac{1}{2}$ or of when

$$odds = \frac{p}{1-p} = \frac{\frac{1}{2}}{1-\frac{1}{2}} = 1,$$

which is the same as

$$\text{logit } p = \log\left(\frac{p}{1-p}\right) = \log 1 = 0.$$

We have $\text{logit } p = \beta_0 + \beta_1 x = 0$ and so we wish to estimate $x = -\beta_0/\beta_1$.

Coronary heart disease data II

The question of interest

We estimate x from our coefficient estimates:

$$\widehat{x} = -\widehat{\beta}_0/\widehat{\beta}_1.$$

So how do we create confidence intervals (and hypothesis tests) for this random quantity $\widehat{x}$ which is the ratio of two approximately normal random variables, namely $\widehat{\beta}_0$ and $\widehat{\beta}_1$?

We will discuss one technique (the delta method) below, which is quite theoretical and compare this with (parametric and non-parametric) bootstrap estimates which are relatively straightforward.

Coronary heart disease data

Theory

Note that $\widehat{x}$ is a random quantity which is the ratio of two asymptotically normal random variables ($\widehat{\beta_0}$ and $\widehat{\beta_1}$). We're not certain what its underlying distribution will be, so obtaining confidence intervals and/or performing hypothesis tests is problematic.

We do know that, approximately,

$$\widehat{\boldsymbol{\beta}} \dot{\sim} \text{Normal}\left(\boldsymbol{\beta},\ \text{Var}\left(\widehat{\boldsymbol{\beta}}\right)\right)$$

where $\boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix}$ and $\text{Var}\left(\widehat{\boldsymbol{\beta}}\right)$ is the covariance matrix for $\widehat{\boldsymbol{\beta}}$.

Coronary heart disease data

Theory

It turns out (applying some statistical theory) that functions of (asymptotically) normal distributions themselves have asymptotically normal distributions as follows:

$$g(\widehat{\boldsymbol{\beta}}) \dot{\sim} \text{Normal}\left(g(\boldsymbol{\beta}),\ \Delta^T \text{Var}\left(\widehat{\boldsymbol{\beta}}\right) \Delta\right)$$

where for our example $g(\boldsymbol{\beta}) = -\beta_0/\beta_1$ and

$$\Delta^T = \left(\frac{\partial g(\boldsymbol{\beta})}{\partial \beta_0},\ \frac{\partial g(\boldsymbol{\beta})}{\partial \beta_1}\right) = \left(-\frac{1}{\beta_1},\ \frac{\beta_0}{\beta_1^2}\right).$$

For those who like this stuff this is an application of Taylor series with a liberal dose of the central limit theorem. This is known as the *delta method* and is also called the *G-prime method*.

Coronary heart disease data

Theory

We can obtain an estimate of $\text{Var}(\widehat{\boldsymbol{\beta}})$, written $\widehat{\text{Var}(\widehat{\boldsymbol{\beta}})}$ say, from the "glm" object chd.fit by

```
vcov(chd.fit)
##               (Intercept)         age
## (Intercept)       1.27809 -0.0265562
## age              -0.02656  0.0005769
```

Oh BTW,

```
nrow(chd.df)
## [1] 43
```

Coronary heart disease data

An aside

Note that this gives us $\text{Var}\left(\widehat{\beta}_0\right)$ and $\text{Var}\left(\widehat{\beta}_1\right)$ as the diagonal elements and $\text{Cov}(\widehat{\beta}_0, \widehat{\beta}_1)$ on the off-diagonal. You've actually seen this before:

```
coef(summary(chd.fit))
```

```
##             Estimate Std. Error z value  Pr(>|z|)
## (Intercept)  -5.2784    1.13053  -4.669 3.026e-06
## age           0.1103    0.02402   4.593 4.364e-06
```

Thus the standard errors for $\widehat{\beta}_0$ and $\widehat{\beta}_1$ are

```
coef(summary(chd.fit))[, "Std. Error"]
```

```
## (Intercept)         age
##     1.13053     0.02402
```

and can also be calculated using `vcov()`:

```
sqrt(diag(vcov(chd.fit)))
```

```
## (Intercept)         age
##     1.13053     0.02402
```

Coronary heart disease data

Back to the story

So let's estimate the variance of $\widehat{X} = -\widehat{\beta}_0/\widehat{\beta}_1$ using the above theory:

```
delta = c(-1/coef(chd.fit)[2],
          coef(chd.fit)[1]/(coef(chd.fit)[2]^2))
(var.xhat <- t(delta) %*% vcov(chd.fit) %*% delta)
```

```
##       [,1]
## [1,] 4.722
```

As we have a way of estimating $\text{Var}(\widehat{X})$ where $\widehat{X} = -\widehat{\beta}_0/\widehat{\beta}_1$, we can get an approximate 95% CI for $x = -\beta_0/\beta_1$ via

$$\widehat{x} \pm 1.96\,\text{se}(\widehat{X}) \quad \text{where} \quad \text{se} = \sqrt{\widehat{\text{Var}}(\widehat{X})}.$$

Coronary heart disease data

Theory-based CI

So the 95% CI for $x = -\beta_0/\beta_1$ is approximately

```
(CI.x <- -coef(chd.fit)[1]/coef(chd.fit)[2] +
         1.96 * c(-1, 1) * c(sqrt(var.xhat)))
## [1] 43.59 52.11
```

Thus our approximate 95% CI for the age at which there's a 50% chance of having CHD is between 43.6 and 52.1 years of age.

That was a lot of hard work which required a bit of maths and distributional theory and depended on several approximations—surely there has to be an easier way.

Coronary heart disease data

Parametric bootstrap

Let's implement parametric bootstrapping. That is, we treat our original estimated coefficients as known parameter values and simulate data from that model to compute a new $\widehat{x} = -\widehat{\beta}_0/\widehat{\beta}_1$ each time.

The original estimated coefficients are:

```
cfs = coef(chd.fit)
cfs
```

```
## (Intercept)         age
##     -5.2784      0.1103
```

Coronary heart disease data

Parametric bootstrap CI

The following code implements the parametric bootstrap, and obtains the approximate 95% CI from the corresponding quantiles of the simulated values of $\widehat{X}$:

```
set.seed(123456)  # For reproducibility of results
Age = chd.df$age
xbeta = cfs[1]+cfs[2]*Age  # Compute Xbeta for each obsn
n = chd.df$n
n.obs = nrow(chd.df)
Nsim = 1e4
betas = matrix(0, nr=Nsim, nc=2)
for (i in 1:Nsim) {
  ysim = rbinom(n.obs, size=n, prob=exp(xbeta)/(1+exp(xbeta)) )
  mod_i = glm(cbind(ysim, n-ysim)~Age, family=binomial)
  betas[i, ] = coef(mod_i)
}
est = -coef(chd.fit)[1] / coef(chd.fit)[2]
c(2*est-quantile(-betas[, 1]/betas[, 2], prob=0.975),
  2*est-quantile(-betas[, 1]/betas[, 2], prob=0.025))
## (Intercept) (Intercept)
##       43.18       51.89
```

Coronary heart disease data

Parametric bootstrap CI

So let's compare these intervals:

- The theory-intense delta method yielded a CI for the age at which there's a 50% chance of having CHD of between 43.6 and 52.1 years of age.
- The parametric bootstrap method yielded a CI for the age at which there's a 50% chance of having CHD of between 43.2 and 51.9 years of age.

Here we see that they are very similar, indicating that the parametric bootstrap method produces a similar result without relying on all of the tricky theory.

For completeness, let's now see how the non-parametric bootstrap compares.

Coronary heart disease data

Non-parametric bootstrap.

Recall, the non-parametric bootstrap simply uses the original data and takes repeated samples from it (with replacement) to compute the statistic of interest. It is particularly useful when your data do not appear to strictly satisfy the distributional assumption of the `"glm"`.

Note for this example, the binomial distribution seems justified indicating that a parametric bootstrap should work well. So we do not anticipate much difference but let's find out if this is the case.

Coronary heart disease data

Non-parametric bootstrap.

We begin by "ungrouping" this grouped data so that we have the original observations (i.e., before we grouped them), from which we need to resample with replacement[2].

```
new.age = rep(chd.df$age, chd.df$n)
length(new.age)
## [1] 100

tmp3 <- rep(c(t(cbind(chd.df$y,
                      chd.df$n-chd.df$y))))
new.y = rep(rep(c(1, 0), nrow(chd.df)), tmp3)
chd.df2 = data.frame(age=new.age, y=new.y)
# Check you get the same parameter estimates as before:
coef(glm(y~age, binomial, data=chd.df2))  # It does
## (Intercept)         age
##     -5.2784      0.1103
```

[2] Can you think of a way of avoiding this step?

Coronary heart disease data

Non-parametric bootstrap.

And now for the non-parametric bootstrap...

```
set.seed(123); n = nrow(chd.df2); Nsim = 1e4
betasBS = matrix(0, Nsim, 2)
for (i in 1:Nsim) {
# Draw a random sample (with replacement) from these data:
  id = sample(1:n, n, replace = TRUE)
  BS.df = chd.df2[id, ]  # Bootstrap sample
  mod_i = glm(y~age, binomial, data = BS.df)
  betasBS[i, ] = coef(mod_i)
}
c(2*est-quantile(-betasBS[, 1]/betasBS[, 2], prob=0.975),
  2*est-quantile(-betasBS[, 1]/betasBS[, 2], prob=0.025))
## (Intercept) (Intercept)
##       43.06       52.11
```

Coronary heart disease data

Comparison of approximate 95% CIs

So let's compare these intervals:

- The theory-intense delta method yielded a CI for the age at which there's a 50% chance of having CHD of between 43.6 and 52.1 years of age.
- The parametric bootstrap method yielded a CI for the age at which there's a 50% chance of having CHD of between 43.2 and 51.9 years of age.
- Here, the non-parametric bootstrap method yielded a CI between 43.1 and 52.1 years of age.

This is very close to the other two CIs.

Coronary heart disease data

Making the theory-based CI easier

Let's see if any fellow nerds have made this easier to use in **R**.
Yep. The delta method can be invoked fairly easily:

```
library("msm")
estmean <- coef(chd.fit)
estvar <- vcov(chd.fit)
dm.sd <- deltamethod (~x1 / x2, estmean, estvar)
dm.sd^2
## [1] 4.722
(CI.x <- -coef(chd.fit)[1]/coef(chd.fit)[2] + 1.96 * c(-1, 1) * dm.sd)
## [1] 43.59 52.11
```

So, at the end of the day the theoretical method is easier. . . but it relies on asymptotic normality.

Macrorhabdus ornithogaster chicken data

Chickens come home to roost

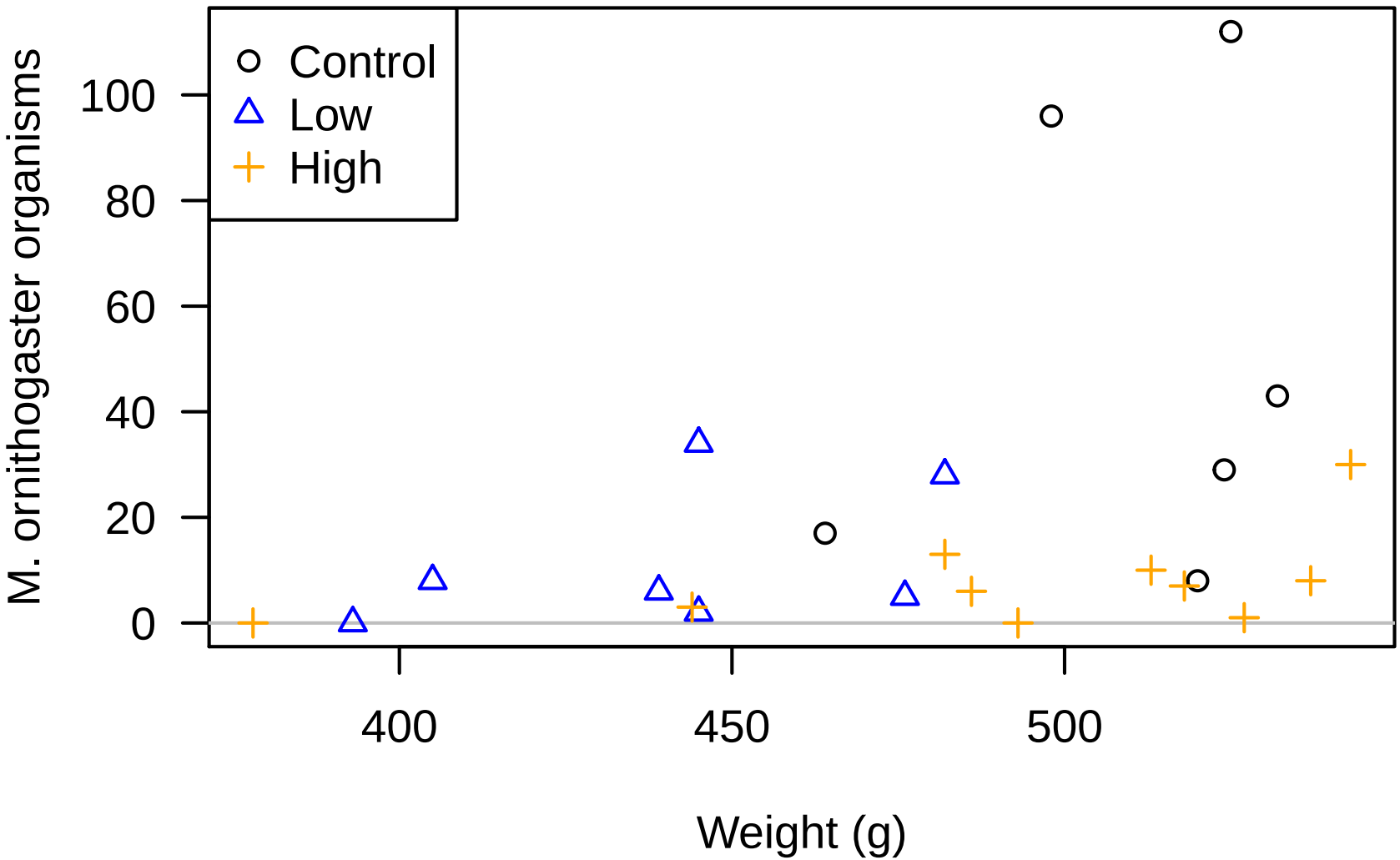
M. ornithogaster is a bacterial disease that affects birds. The drug Amphotericin B is often used for treatment, but its efficacy has not yet been established. It was of interest to determine if the Amphotericin B dose is related to the number of *M. ornithogaster* organisms shed in chickens' faeces, while correcting for the size of the chicken. An experiment was conducted on 23 infected chickens.
The data frame contains the following variables:

- `mo`: The number of *M. ornithogaster* in a sample of ten faecal slides from the chicken
- `dose`: The dose of Amphotericin B given to the chicken; either `control`, `low`, or `high`
- `weight`: The weight of the chicken in grams

You will recall the relationship between `dose`, `weight` and `mo`.

Macrorhabdus ornithogaster chicken data

The data, with dose groups



Macrorhabdus ornithogaster chicken data

An additive model assuming a Poisson distribution:

```
chickens.fit = glm(mo~weight+dose, poisson, data=chickens.df)
summary(chickens.fit)
```

```
## Call:
## glm(formula = mo ~ weight + dose, family = poisson, data = chickens.df)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.57817    1.01576   -2.54  0.01114 *
## weight       0.01267    0.00196    6.45  1.1e-10 ***
## doseHigh    -1.74362    0.12718  -13.71  < 2e-16 ***
## doseLow     -0.60474    0.17709   -3.41  0.00064 ***
## ---
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 683.02  on 22  degrees of freedom
## Residual deviance: 310.61  on 19  degrees of freedom
```

We found that we have very strong reason to believe that we have overdispersion as $\Pr\left(\chi^2_{df=19} > 310.61\right) \approx 0$.

Macrorhabdus ornithogaster chicken data

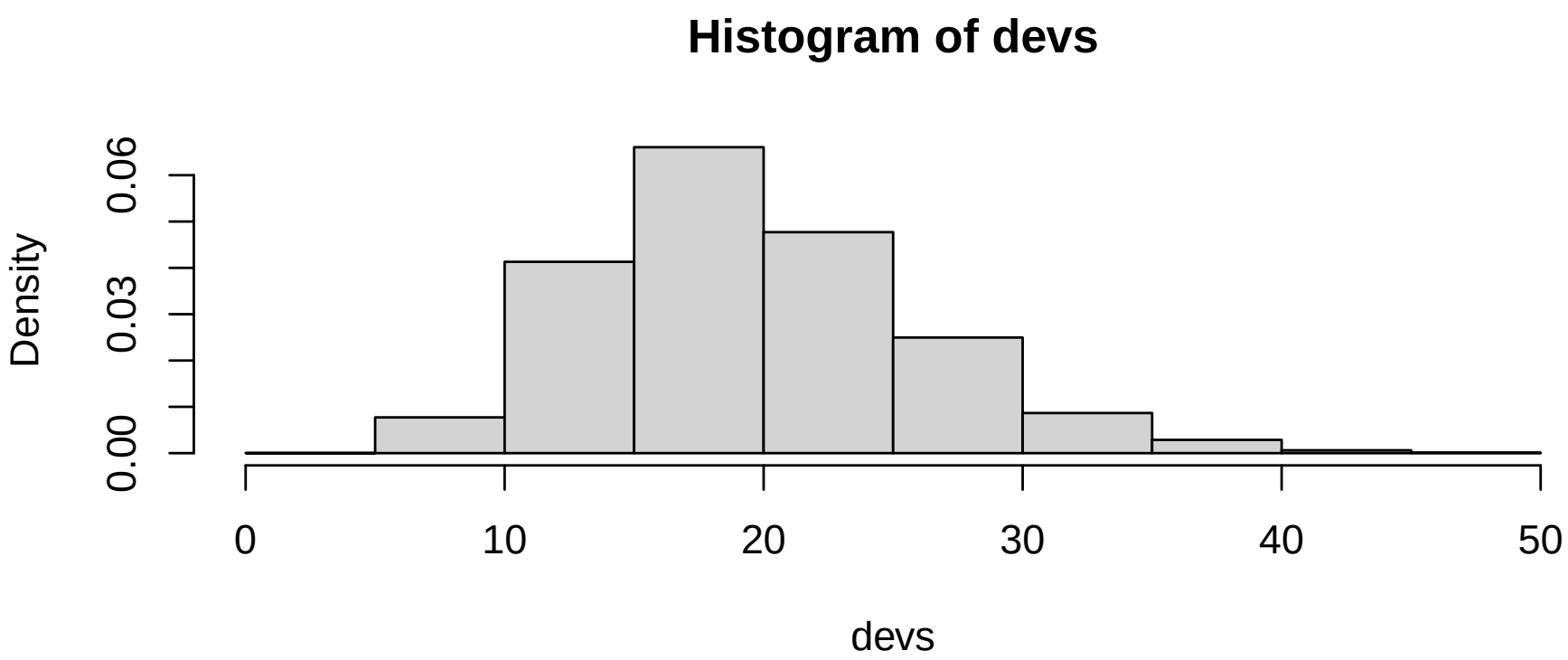
Let's do a parametric bootstrap to check whether the distribution of the deviance is well approximated by the $\chi^2_{df=19}$ distribution.

Remember, we use the fitted Poisson model as the true relationship to repeatedly simulate data:

```
Nsim = 1e4
devs = numeric(Nsim)
nr = nrow(chickens.df)
for (i in 1:Nsim) {
  preds = predict(chickens.fit, type="response")
  mo_sim = rpois(nr, preds)
  devs[i] = deviance(glm(mo_sim~dose+weight, poisson, data = chickens.df))
}
```

Macrorhabdus ornithogaster chicken data

```
hist(devs, freq = FALSE)
```



The parametric bootstrap confirms that the observed deviance is not plausible if the data come from the Poisson model. So, let's try the negative binomial model.

Macrorhabdus ornithogaster chicken data

Negative binomial model

```
nbChickens.fit = glm.nb(mo~weight+dose, data=chickens.df)
slimSummary(nbChickens.fit)
```

```
## Call:
## glm.nb(formula = mo ~ weight + dose, data = chickens.df, init.theta = 1.2767689,
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -5.7203     3.1894   -1.79   0.0729 .
## weight        0.0189     0.0062    3.05   0.0023 **
## doseHigh     -1.8083     0.4861   -3.72   0.0002 ***
## doseLow      -0.2892     0.6630   -0.44   0.6627
## ---
## (Dispersion parameter for Negative Binomial(1.277) family taken to be 1)
##
##     Null deviance: 53.183  on 22  degrees of freedom
## Residual deviance: 26.408  on 19  degrees of freedom
```

```
nb.dev = deviance(nbChickens.fit)
nb.df = nbChickens.fit$df.resid
cat("GOF chi-square p-value for negbin model is", 1-pchisq(nb.dev, nb.df))
```

```
## GOF chi-square p-value for negbin model is 0.1192
```

Macrorhabdus ornithogaster chicken data

Negative binomial deviance check

Although not significant at the 5% level, the residual deviance of 26.41 is a bit higher than we would like under the $\chi^2_{df=19}$ approximation. Let's implement a parametric bootstrap to check the sampling distribution of the deviance[3]

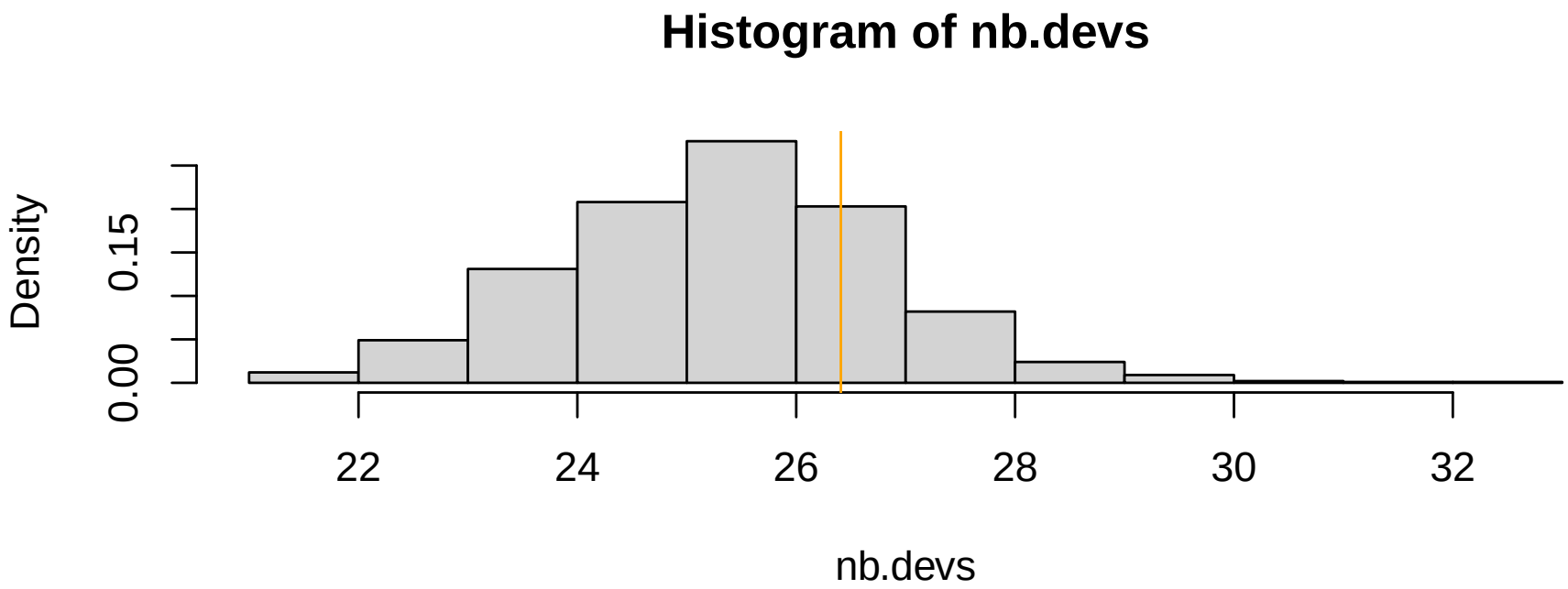
```
set.seed(12345); Nsim = 1e3  # 1e4 Takes too long!
nb.devs = numeric(Nsim); nr = nrow(chickens.df)
for (i in 1:Nsim) {
  mos_nb = rnbinom(nr, mu = predict(nbChickens.fit, type="resp"),
                   size = nbChickens.fit$theta)
  mod_i = glm.nb(mos_nb~weight+dose, data=chickens.df)
  nb.devs[i] = deviance(mod_i)
}
cat("GOF parametric bootstrap p-value for NB model is",
    1-mean(nb.devs < nb.dev))
## GOF parametric bootstrap p-value for NB model is 0.238
```

[3]Note: careful examination of the NB fits showed that some fitted models did not converge. Hence the results need to be taken with caution.

Macrorhabdus ornithogaster chicken data

Negative binomial deviance check

```
hist(nb.devs, freq = FALSE)
abline(v = deviance(nbChickens.fit), col = "orange")
```



Under the negative binomial model, the sampling distribution of the residual deviance is not well approximated by a $\chi^2_{df=19}$ distribution. Why?

A Bit about Convergence

Occasionally, we may have a problem with getting `glm()` / `glm.nb()` to converge.

- Lack of convergence may be an indication that the fitted model is not stable (i.e., a small change in the data results in a big change in the estimated coefficients).
- More apt to occur when the number of observations is small and/or the model is complicated.
- The non-parametric bootstrap can exacerbate this problem since a bootstrap sample omits (on average) 37% of the data.
- If `glm()` / `glm.nb()` doesn't converge, you can try increasing the maximum number of iterations using the `maxit` argument.

Conclusion

Deja vu from earlier

Bootstrapping allows the sampling distribution of almost any statistic to be estimated empirically. It is particularly useful in cases where:

- It is difficult or impossible to obtain a sampling distribution based on theory.
- The assumptions required to obtain a theoretical sampling distribution are in doubt.

Some important points:

- The bootstrap is NOT used to get a better estimate of a statistic—the expected value of the bootstrap estimate is equal to the sample estimate.
- The data must be independent and representative of the population.
- Does not compensate for model misspecification.