Introduction to statistics: Linear models

Shravan Vasishth

Universität Potsdam
vasishth@uni-potsdam.de
http://www.ling.uni-potsdam.de/∼vasishth

April 12, 2020

Summary

1. We learnt about the single sample, two sample, and paired t-tests.
2. We learnt about Type I, II error (and power).
3. We learnt about Type M and Type S errors.

Now we are ready to look at linear modeling.

Load Grodner and Gibson dataset

```
gge1crit<-read.table("data/grodnergibson05data.txt",header=
head(gge1crit)

##     subject item condition rawRT
## 6         1    1    objgap   320
## 19        1    2   subjgap   424
## 34        1    3    objgap   309
## 49        1    4   subjgap   274
## 68        1    5    objgap   333
## 80        1    6   subjgap   266
```

Compute means by factor level

Let's compute the means by factor levels:

```
means<-round(with(gge1crit,tapply(rawRT,IND=condition,
                                  mean)))
means

##  objgap subjgap
##     471     369
```

The object relative mean is higher than the subject relative mean.

Paired t-test on the data

Correct t-test by subject and by items

This is how one would do a t-test CORRECTLY with such data, to compare means across conditions:

```
bysubj<-aggregate(rawRT~subject+condition,
                  mean,data=gge1crit)
byitem<-aggregate(rawRT~item+condition,mean,data=gge1crit)
t.test(rawRT~condition,paired=TRUE,bysubj)$statistic

##      t
## 3.1093

t.test(rawRT~condition,paired=TRUE,byitem)$statistic

##      t
## 3.7542
```

Paired t-test on the data

Consider only by-subject analyses for now.
These are the means we are comparing by subject:

```
round(with(bysubj,
tapply(rawRT,condition,mean)))

##  objgap subjgap
##     471     369
```

Linear models

We can rewrite our best guess about how the object and subject relative clause reading time distributions like this:
Object relative: $Normal(471, \hat{\sigma})$
Subject relative: $Normal(471 - 102, \hat{\sigma})$
Note that the two distributions for object and subject relative are assumed to be independent. This is not true in our data as we get a data point each for each RC type from the same subject!

Linear models

- The object relative's distribution can be written as a sum of two terms:
  $y = 471 + \epsilon$ where $\epsilon \sim Normal(0, \hat{\sigma})$
- The subject relative's distribution can be written:
  $y = 471 - 102 + \epsilon$ where $\epsilon \sim Normal(0, \hat{\sigma})$
- Note that $\hat{\sigma} = 213$ because $obs.t = \frac{\bar{x}}{s/\sqrt{n}} \Rightarrow s = \bar{x} \times \sqrt{n}/obs.t = -103 \times \sqrt{42}/-3.109 = 213.$

The above statements describe a generative process for the data.

Linear models

Now consider this **linear model**, which describes the rt in each row of the data frame as a function of condition. ϵ is a random variable $\epsilon \sim Normal(0, 213)$.
Object relative reading times:

$$rt = 471 + \epsilon \tag{1}$$

Subject relative reading times:

$$rt = 471 - 102 + \epsilon \tag{2}$$

Linear models

When describing mean reading times, I can drop the ϵ:
Object relative reading times:

$$rt = 471 \tag{3}$$

Subject relative reading times:

$$rt = 471 - 102 \tag{4}$$

The lm() function gives us these mean estimates from the data.

Linear models

Object relative reading times:

$$rt = 471 \times \mathbf{1} - 102 \times \mathbf{0} + \epsilon \tag{5}$$

Subject relative reading times:

$$rt = 471 \times \mathbf{1} - 102 \times \mathbf{1} + \epsilon \tag{6}$$

So, object relatives are coded as 0, and subject relatives are coded as 1.
The lm() function sets up such a model.

Linear models

With real data from the relative clause study:

```
contrasts(bysubj$condition)

##         subjgap
## objgap        0
## subjgap       1

m0<-lm(rawRT~condition,bysubj)
round(summary(m0)$coefficients)[,1]

##      (Intercept) conditionsubjgap
##              471              -102
```

Linear models

The linear model gives us two numbers: object relative reading time (471), and the difference between object and subject relative (-102):

```
round(coef(m0))
```

```
##       (Intercept) conditionsubjgap
##               471              -102
```

Linear models

1. The **intercept** is giving us the mean of the objgap condition.
2. The **slope** is giving us the amount by which the subject relative is faster.

Note that the meaning of the intercept and slope depends on the ordering of the factor levels. We can make subject relative means be the intercept:

```
## reverse the factor level ordering:
bysubj$condition<-factor(bysubj$condition,
                         levels=c("subjgap","objgap"))
contrasts(bysubj$condition)

##         objgap
## subjgap      0
## objgap       1
```

Linear models

```
m1a<-lm(rawRT~condition,bysubj)
round(coef(m1a))

##     (Intercept) conditionobjgap
##             369             102
```

Now the intercept is the subject relative clause mean.
The slope is the increase in reading time for the object relative condition.

Linear models

```
## switching back to the original
## factor level ordering:
bysubj$condition<-factor(bysubj$condition,
                         levels=c("objgap","subjgap"))
contrasts(bysubj$condition)

##         subjgap
## objgap        0
## subjgap       1
```

Linear models

In mathematical form, the model is:

$$rt = \beta_0 + \beta_1 condition + \epsilon \tag{7}$$

where

- β_0 is the mean for the object relative
- β_1 is the amount by which the object relative mean must be changed to obtain the mean for the subject relative.

The null hypothesis is that the difference in means between the two relative clause types β_1 is:
$H_0 : \beta_1 = 0$

Linear models

The **contrast coding** determines the meaning of the β parameters:

```
bysubj$condition<-factor(bysubj$condition,
                         levels=c("objgap","subjgap"))
contrasts(bysubj$condition)

##         subjgap
## objgap        0
## subjgap       1
```

Linear models

We will make a distinction between the **unknown true mean** β_0, β_1 and the **estimated mean from the data** $\hat{\beta}_0, \hat{\beta}_1$.

- Estimated mean object relative processing time: $\hat{\beta}_0 = 471$.
- Estimated mean subject relative processing time: $\hat{\beta}_0 + \hat{\beta}_1 = 471 + -102 = 369$.

Linear models

Reparameterizing the linear model with sum contrast coding

In mathematical form, the model is:

$$rt = \beta_0 + \beta_1 condition + \epsilon \tag{8}$$

We can change the **contrast coding** to change the meaning of the β parameters:

```
## new contrast coding:
bysubj$cond<-ifelse(bysubj$condition=="objgap",1,-1)
```

Linear models

Reparameterizing the linear model with sum contrast coding

```
xtabs(~cond+condition,bysubj)

##      condition
## cond objgap subjgap
##   -1      0      42
##   1      42       0
```

Linear models

Reparameterizing the linear model with sum contrast coding

Now the model parameters have a different meaning:

```
m1<-lm(rawRT~cond,bysubj)
round(coef(m1))

## (Intercept)        cond
##         420          51
```

Linear models

Reparameterizing the linear model with sum contrast coding

- Estimated **grand mean** processing time: $\hat{\beta}_0 = 420$.
- Estimated mean object relative processing time: $\hat{\beta}_0 + \hat{\beta}_1 = 420 + 51 = 471$.
- Estimated mean subject relative processing time: $\hat{\beta}_0 - \hat{\beta}_1 = 420 - 51 = 369$.

This kind of parameterization is called **sum-to-zero contrast** or more simply **sum contrast** coding. This is the coding we will use.

Linear models

The null hypothesis for the slope

The null hypothesis for the slope is

$$H_0 : \mathbf{1} \times \mu_{obj} + (-\mathbf{1} \times) \mu_{subj} = 0 \tag{9}$$

The sum contrasts are referring to the ± 1 terms in the null hypothesis:

- object relative: $+1$
- subject relative: -1

Linear models

Now the model is:
Object relative reading times:

$$rt = 420 \times \mathbf{1} + 51 \times \mathbf{1} + \epsilon \tag{10}$$

Subject relative reading times:

$$rt = 420 \times \mathbf{1} + 51 \times -\mathbf{1} + \epsilon \tag{11}$$

So, object relatives are coded as 1, and subject relatives are coded as -1.

Linear models

The model is:

$$rt = \beta_0 + \beta_1 + \epsilon \text{ where } \epsilon \sim Normal(0, \sigma) \tag{12}$$

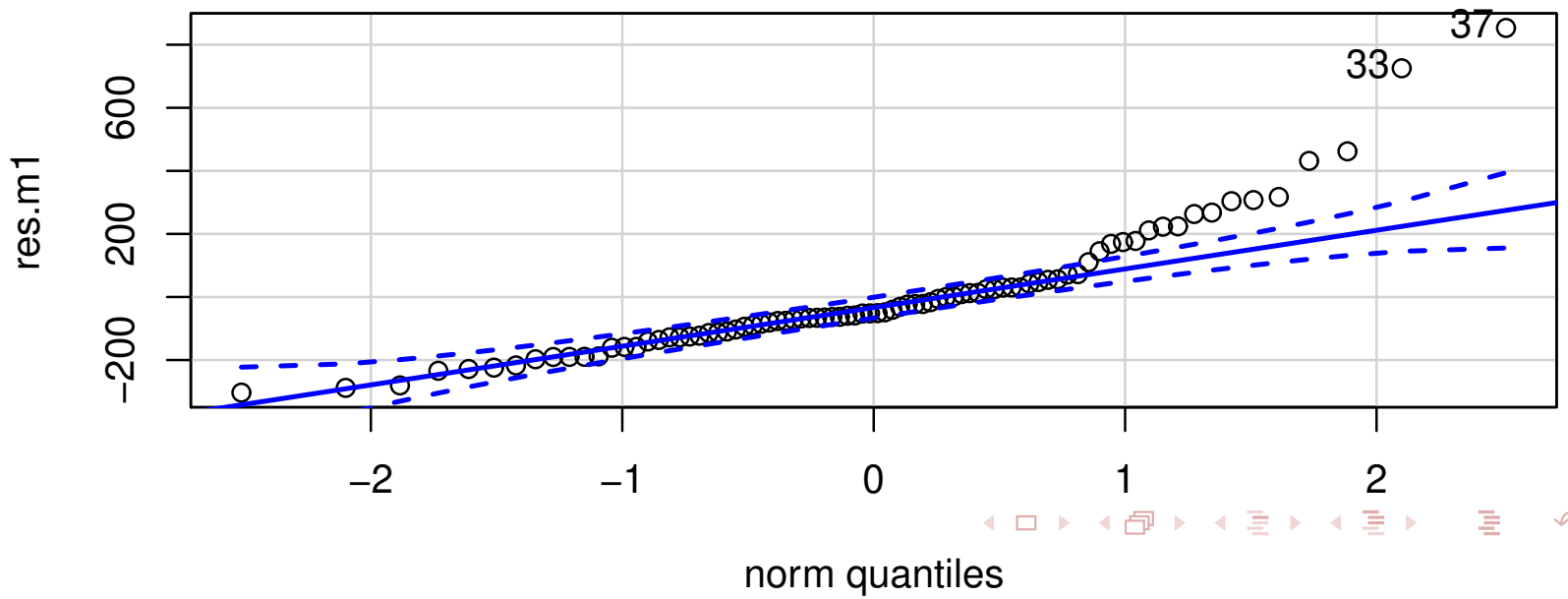
It is an assumption of the linear model that the residuals are (approximately) normally distributed.
We can check that this assumption is met:

```
## residuals:
res.m1<-residuals(m1)
```

Linear models

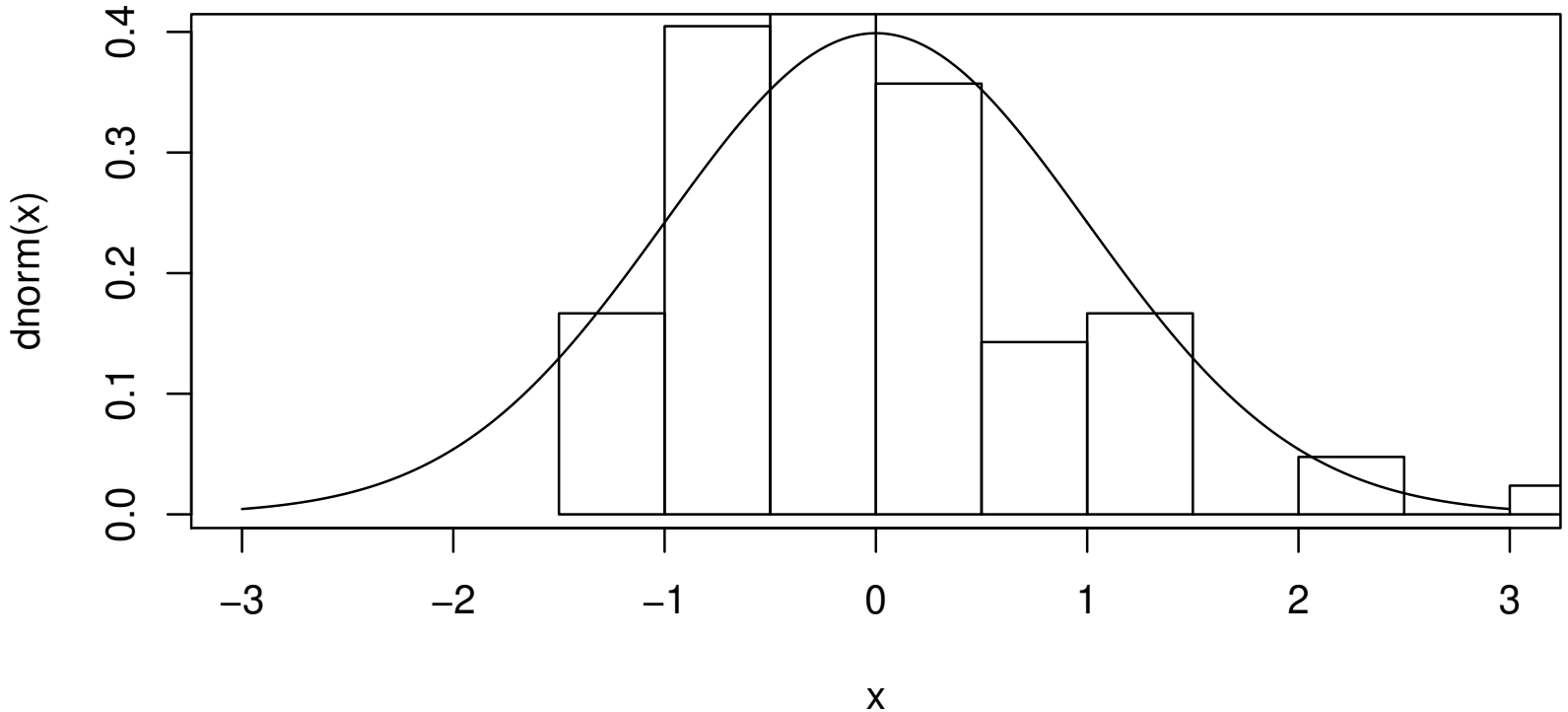
Plot the residuals by comparing them to the standard normal distribution (Normal(0,1)):

```
## [1] 37 33
```



Linear models

Another way to visualize it (not the standard way):



Linear models

Log transformation

A log-transform improves the normality of residuals:

```
m1log<-lm(log(rawRT)~cond,bysubj)
round(coef(m1log),4)

## (Intercept)        cond
##      5.9488      0.0843
```

Linear models

Log transformation

- Estimated **grand mean** processing time: $\hat{\beta}_0 = 5.9488$.
- Estimated mean object relative processing time: $\hat{\beta}_0 + \hat{\beta}_1 = 5.9488 + 0.0843 = 6.0331$.
- Estimated mean subject relative processing time: $\hat{\beta}_0 - \hat{\beta}_1 = 5.9488 - 0.0843 = 5.8645$.

Linear models

The model is:

$$\log rt = \beta_0 + \beta_1 + \epsilon \tag{13}$$

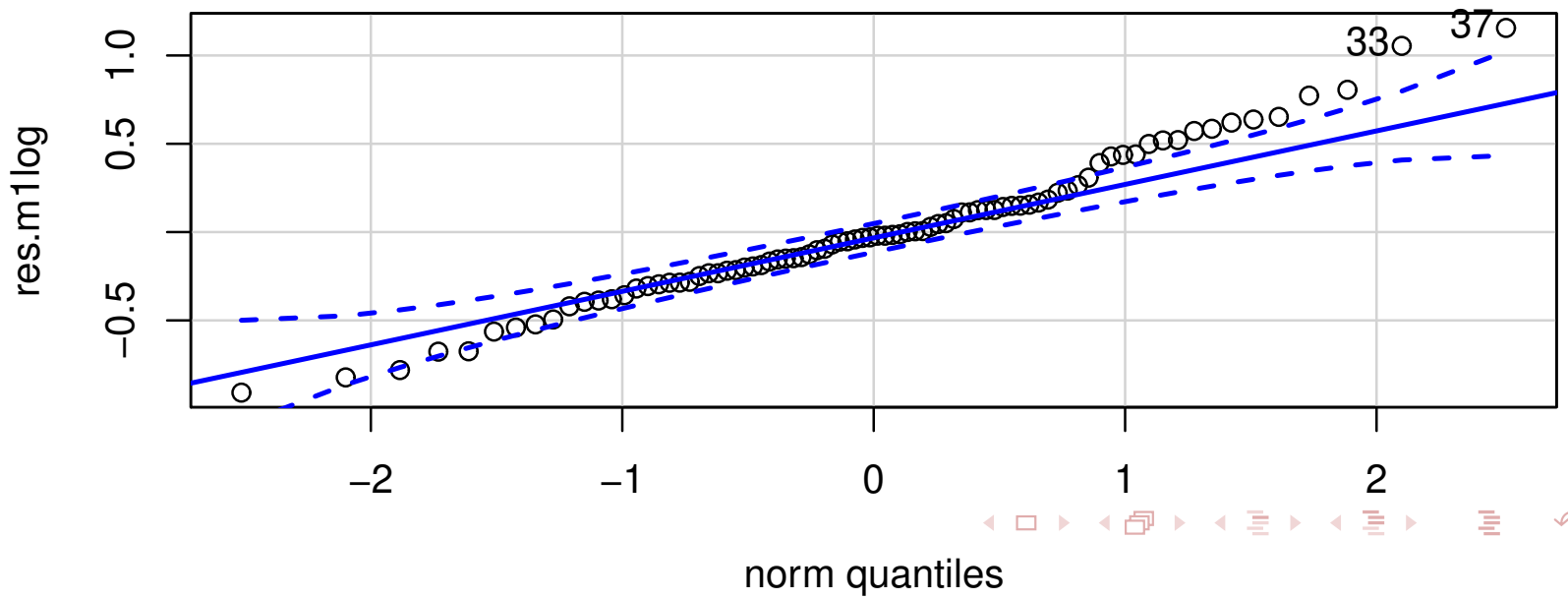
Now check the residuals:

```
## residuals:
res.m1log<-residuals(m1log)
```

Linear models

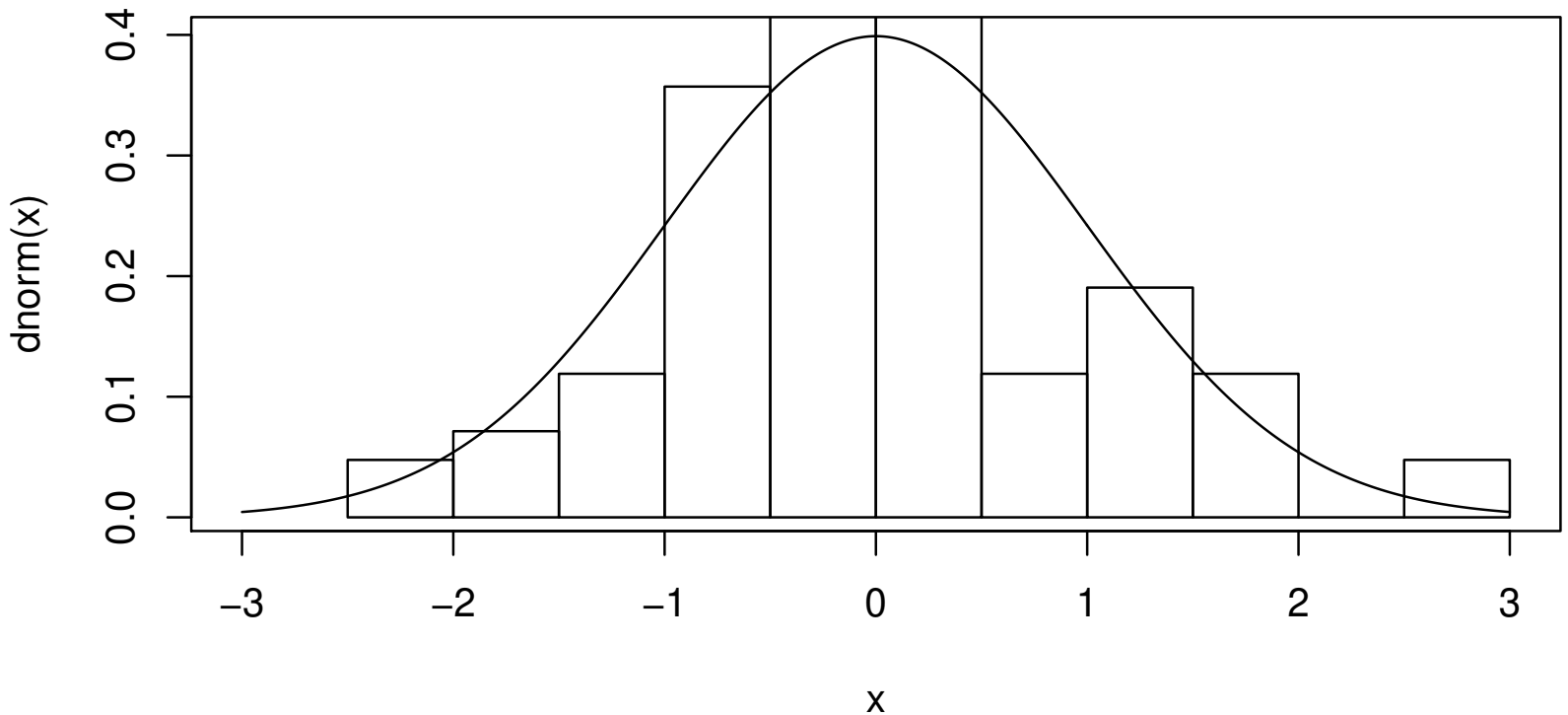
Plot the residuals by comparing them to the standard normal distribution (Normal(0,1)):

```
## [1] 37 33
```



Linear models

Another way to visualize it (not the standard way):



Linear models

Log transformation: recovering estimates on ms scale

- Estimated mean object relative processing time:
  $\hat{\beta}_0 + \hat{\beta}_1 = 5.9488 + 0.0843 = 6.0331.$
- Estimated mean subject relative processing time:
  $\hat{\beta}_0 - \hat{\beta}_1 = 5.9488 - 0.0843 = 5.8645.$

Note that $exp(log(rt)) = rt$.
To get the mean estimates on the raw ms scale, we just exponentiate both sides of the equation:
$exp(\log rt) = exp(\beta_0 + \beta_1)$

Linear models

Log transformation: recovering estimates on ms scale

- Estimated mean object relative processing time:
  $exp(\hat{\beta}_0 + \hat{\beta}_1) = exp(5.9488 + 0.0843) = 417.$
- Estimated mean subject relative processing time:
  $exp(\hat{\beta}_0 - \hat{\beta}_1) = exp(5.9488 - 0.0843) = 352.$

The difference in reading time is 417-352=65 ms (cf. 102 ms on raw scale).

Linear models: Summary

Summary

Using the relative clause example, we learnt

- the meaning of treatment contrast coding.
- the meaning of sum contrast coding. **This is the coding we will use**.

For a more comprehensive discussion of contrast coding, read this paper:
How to capitalize on a priori contrasts in linear (mixed) models: A tutorial, Schad, Hohenstein, Vasishth, Kliegl. Download from: https://arxiv.org/abs/1807.10451

Paired t-tests vs linear models

The linear mixed model

The paired t-test and the linear model t-test values don't match:

```
t.test(rawRT~condition,bysubj,paired=TRUE)$statistic
```

```
##      t
## 3.1093
```

```
round(summary(m0)$coefficients,2)[,c(1:3)]
```

```
##                   Estimate Std. Error t value
## (Intercept)         471.36      31.13   15.14
## conditionsubjgap   -102.29      44.02   -2.32
```

Paired t-tests vs linear models

The linear mixed model

This is because the linear model implements the **unpaired** (i.e., two sample) t-test:

```
round(summary(m0)$coefficients,2)[,c(1:3)]

##                   Estimate Std. Error t value
## (Intercept)         471.36      31.13   15.14
## conditionsubjgap   -102.29      44.02   -2.32

round(t.test(rawRT~condition,bysubj,
             paired=FALSE)$statistic,2)

##    t
## 2.32
```

Paired t-tests vs linear models

The linear mixed model

The paired t-test has an equivalent in the linear modeling framework:

```
m0.lmer<-lmer(rawRT~condition+(1|subject),bysubj)
summary(m0.lmer)$coefficients

##                   Estimate Std. Error t value
## (Intercept)         471.36     31.128 15.1428
## conditionsubjgap   -102.29     32.896 -3.1093
```

We turn to the linear mixed model in the next lecture.

Linear models

1. In our relative clause example, the 'predictor' is categorial.
2. What about when we have continuous predictors?
3. For example, we have instructors' "beauty" levels measured on a continuous scale as predictors of their teaching evaluations.
4. Beauty levels are centered; this means that a beauty level of 0 means average beauty level. This is a data set from a paper by Hamermesh and Parker (Beauty in the Classroom: Instructors' Pulchritude and Putative Pedagogical Productivity," Economics of Education Review, August 2005). I got the data from Gelman and Hill (2007).

Linear models

```
bdata <- read.table("data/beauty.txt",header=T)
head(bdata)

##     beauty evaluation
## 1  0.20157        4.3
## 2 -0.82608        4.5
## 3 -0.66033        3.7
## 4 -0.76631        4.3
## 5  1.42145        4.4
## 6  0.50022        4.2

plot(evaluation~beauty,bdata)
```

Linear models

Note that the beauty scores are centered to have mean (approximately) 0:

```
summary(bdata$beauty)

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1.5388 -0.7446 -0.1564 -0.0883  0.4572  1.8817
```

Linear models

One model we can fit:

$y = \beta_0 + \epsilon$

```
m2<-lm(evaluation~1,bdata)
mean(bdata$evaluation)

## [1] 3.9983

round(summary(m2)$coefficients,digits=3)

##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    3.998      0.026  155.05        0
```

This model is only estimating the grand mean of evaluation scores.

Linear models

$$y = \beta_0 + \beta_1 x + \epsilon$$

```
m3<-lm(evaluation~beauty,bdata)
round(summary(m3)$coefficients,digits=3)

##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    4.010      0.026 157.205        0
## beauty         0.133      0.032   4.133        0
```

The intercept now means: the expected evaluation score given an average beauty level.
The slope means: the expected increase in evaluation with **one unit** increase in beauty.

Summary

We now know how to fit

1. Simple linear models with a categorical predictor (relative clause data)
2. Simple linear models with a continuous predictor.