Statistical methods for linguistic research: Foundational Ideas

Shravan Vasishth

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

August 4, 2015

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.

Maxwell and Delaney dataset

1. Here's a simplified data set from Maxwell and Delaney's book (Designing Expts and Analyzing Data).
2. This is **within-subjects** (fake) experimental data; it has a 2×2 design.
3. Within-subjects designs have the property that each subject sees every condition in the experiment.
4. Contrast this to **between**-subjects designs, e.g., medical trials with treatment vs placebo conditions.

Maxwell and Delaney dataset

Imagine that subjects are shown a stimulus (a picture) on the screen, and it's either shown with no noise (distortion, say) in the background, or with noise; in addition, the stimulus was either horizontal (tilted 0 degrees), or tilted by 4 degrees.
The experiment has two levels of noise, and two levels of tilt.

Load the data

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

##    rt subj deg    noise
## 1 420   s1   0 no.noise
## 2 420   s2   0 no.noise
## 3 480   s3   0 no.noise
## 4 420   s4   0 no.noise
## 5 540   s5   0 no.noise
## 6 360   s6   0 no.noise
```

Load the data

Let's compute the means by factor levels:

```
means<-with(noisedeg,tapply(rt,IND=list(noise,deg),
                            mean))
means

##              0   4
## no.noise 462 510
## noise    492 660
```

Paired t-test on the data

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

```
no.noise<-subset(noisedeg,noise=="no.noise")
no.noise.means<-with(no.noise,tapply(rt,subj,mean))

noise<-subset(noisedeg,noise=="noise")
noise.means<-with(noise,tapply(rt,subj,mean))
```

Paired t-test on the data

```
no.noise.means

##  s1 s10  s2  s3  s4  s5  s6  s7  s8  s9
## 420 450 450 480 480 600 390 480 540 570

noise.means

##  s1 s10  s2  s3  s4  s5  s6  s7  s8  s9
## 540 600 420 720 630 570 420 630 630 600
```

Paired t-test on the data

```
t.test(noise.means,no.noise.means,paired=TRUE)

##
##  Paired t-test
##
## data:  noise.means and no.noise.means
## t = 3.2225, df = 9, p-value = 0.01045
## alternative hypothesis: true difference in means is not
## 95 percent confidence interval:
##   26.82139 153.17861
## sample estimates:
## mean of the differences
##                      90
```

Paired t-test on the data

Note that the order in which we compare the two vectors does not matter:

```
t.test(noise.means,no.noise.means,
       paired=TRUE)$statistic

##         t
## 3.222517

t.test(no.noise.means,noise.means,
       paired=TRUE)$statistic

##          t
## -3.222517
```

All that changes is the sign of the t-value.

Paired t-test on the data

These are the means we are comparing in the noise case:

```
## means of noise levels:
with(noisedeg,tapply(rt,noise,mean))

## no.noise    noise
##      486      576

## mean of no.noise=500
## mean of noise=500+138=638
```

Paired t-test on the data

And here are the means we would compare in the degree case if we were to do pairwise t-tests on the different levels (0 vs 4 deg):

```
with(noisedeg,tapply(rt,deg,mean))

##   0   4
## 477 585

## mean of 0 deg=477
## mean of 4 deg=477+108=585
```

As for noise, we could fit t-tests repeatedly for degree as well. (Try this yourself later.)

Linear models

Now consider this **linear model**, which evaluates rt as a function of noise (**Note: this model is incorrect for the present dataset, but it's useful to understand how it work in order to explain how linear mixed models work**):

```
m0<-lm(rt~noise,noisedeg)
round(summary(m0)$coefficients,digits=2)

##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)      486      23.70   20.50     0.00
## noisenoise        90      33.52    2.68     0.01
```

Linear models

It's important to understand what lies behind this output.
We won't go through everything in the output, but only focus on some aspects that are immediately of relevance to us. First, we have the **coefficients**, the intercept and slope of the fitted line respectively:

```
coef(m0)

## (Intercept)  noisenoise
##         486          90
```

Linear models

One instructive exercise is to compare these coefficients with the means we have for noise. The mean of no.noise is 486, and the mean of noise is 486+90=576:

```
## means of noise levels:
with(noisedeg,tapply(rt,noise,mean))

## no.noise    noise
##      486      576

## mean of no.noise=486
## mean of noise=486+90=576
```

Linear models

1. The intercept is giving us the mean of the no-noise condition.
2. The slope is giving us the difference in means between no-noise and noise.

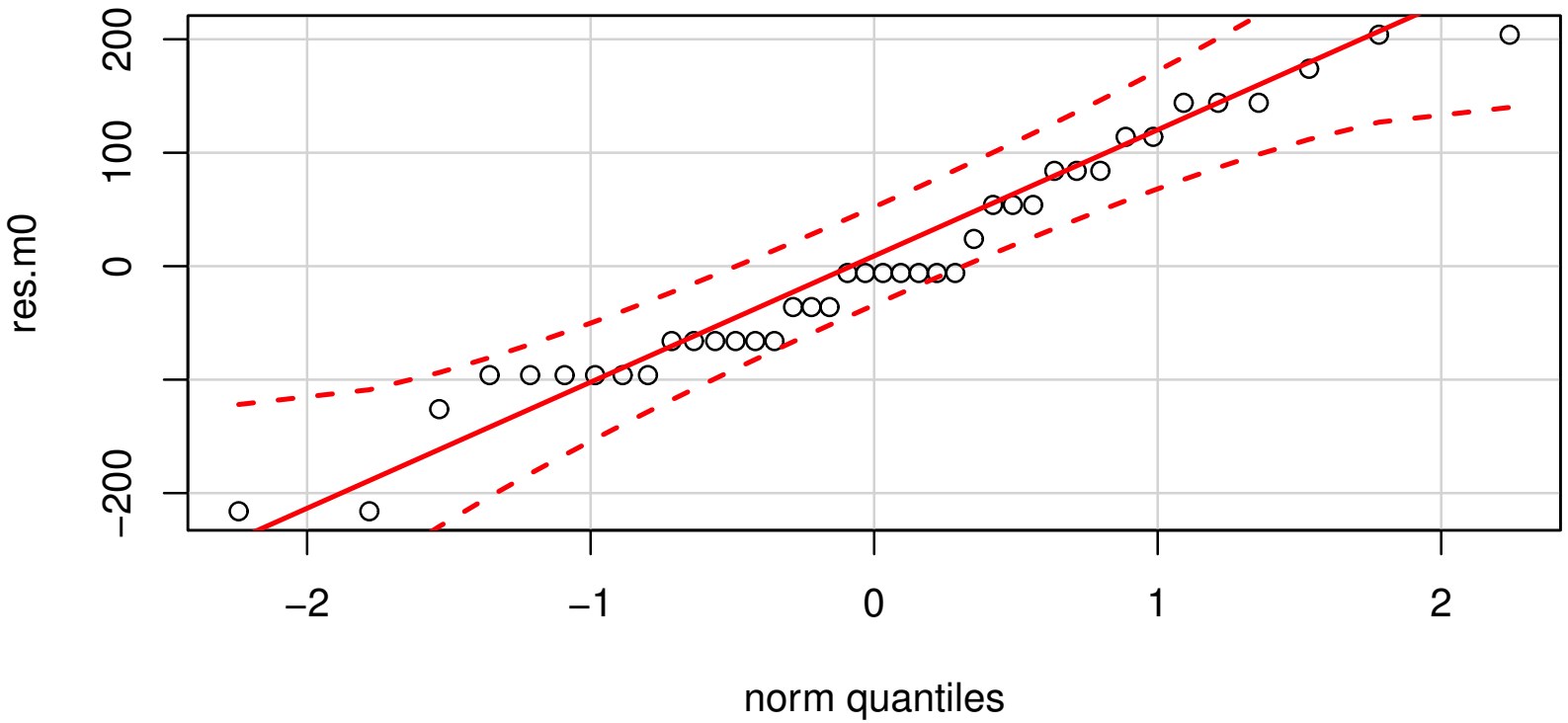
Linear models

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.m0<-residuals(m0)
```

Linear models

Plot the residuals by comparing them to a normal distribution:



Linear models

Next, we will look at the matrix formulation of the linear model.

Linear models

Underlyingly, we have a design matrix or model matrix, which is being used by R to estimate the coefficients:

```
head(model.matrix(m0),n=7)

##   (Intercept) noisenoise
## 1           1          0
## 2           1          0
## 3           1          0
## 4           1          0
## 5           1          0
## 6           1          0
## 7           1          0
```

Linear models

Our linear model equation for the noise model m0 above is a system of equations. The single equation:

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i \tag{1}$$

can be expanded to:

$$\begin{array}{rcccccc} Y_1 & = & \beta_0 & + & X_1\beta_1 & + & \epsilon_1 \\ Y_2 & = & \beta_0 & + & X_2\beta_1 & + & \epsilon_2 \\ Y_3 & = & \beta_0 & + & X_3\beta_1 & + & \epsilon_3 \\ Y_4 & = & \beta_0 & + & X_4\beta_1 & + & \epsilon_4 \\ \vdots & & \vdots & & \vdots & & \vdots \\ Y_n & = & \beta_0 & + & X_n\beta_1 & + & \epsilon_n \end{array} \tag{2}$$

Linear models

And this system of linear equations can be restated in compact matrix form:

$$\mathbf{Y} = \mathbf{X}\beta + \epsilon \tag{3}$$

where

Vector of responses:

$$\mathbf{Y} = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ \vdots \\ Y_n \end{pmatrix} \tag{4}$$

Linear models

The design matrix (in R this is called the model matrix):

$$\mathbf{X} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ \vdots & \vdots \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} \tag{5}$$

Linear models

Vector of parameters to be estimated:

$$\beta = \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} \tag{6}$$

and
Vector of error terms (residuals):

$$\epsilon = \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \vdots \\ \epsilon_n \end{pmatrix} \tag{7}$$

Linear models

We could write the whole equation as:

$$\begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ \vdots \\ Y_n \end{pmatrix} = \begin{pmatrix} 1 & X_1 \\ 1 & X_2 \\ 1 & X_3 \\ 1 & X_4 \\ \vdots & \vdots \\ 1 & X_n \end{pmatrix} \times \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \vdots \\ \epsilon_n \end{pmatrix} \tag{8}$$

Linear models

1. Our main goal when we fit a linear model is to find estimates of the parameters β_0 and β_1, the intercept and slope respectively.
2. We will call the estimates $\hat{\beta}_0$ and $\hat{\beta}_1$.
3. This can be done by "solving" for the beta's. X is the model matrix, and X$'$ is the transpose of the model matrix. Y is the vector of dependent variables.

Linear models

$$\beta = (X'X)^{-1}X'Y \tag{9}$$

You do not need to know how the above equation comes about; all you need to know is that given the design matrix X, and the data Y, we can estimate the parameters.
In case you are interested in more detail, see the derivation these lecture notes: https://github.com/vasishth/LM

Linear models

Contrast coding

Returning to our noise and deg(ree) example above, we can also fit a linear model where we examine the effect of deg on rt.
Here, a critical thing to attend to is the **contrast coding** for the factor degree:

```
noisedeg$deg<-factor(noisedeg$deg)
contrasts(noisedeg$deg)

##   4
## 0 0
## 4 1
```

The above contrast coding says the following: compare 0 with deg 4. This kind of contrast is called treatment contrast coding: there's always a baseline that you compare another condition with.

Linear models

Let's fit the model:

```
## evaluating effect of degree:
round(summary(m1<-lm(rt~deg,noisedeg))$coefficients,digits=

##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)      477     22.691  21.021    0.000
## deg4             108     32.090   3.365    0.002

## compare the coefficients with means computed earlier
```

Compare the coefficients with these means for deg:

```
with(noisedeg,tapply(rt,deg,mean))

##   0   4
## 477 585
```

Linear models

We can also examine the effects of noise and degree together:

```
## evaluating effect of noise AND degree together:
m2<-lm(rt~noise+deg,noisedeg)
round(summary(m2)$coefficients,digits=3)

##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)      432      25.08  17.225    0.000
## noisenoise        90      28.96   3.108    0.004
## deg4             108      28.96   3.729    0.001
```

Linear models

1. In our current noise and deg example, the 'predictors' are categorial ones.
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  0.2015666        4.3
## 2 -0.8260813        4.5
## 3 -0.6603327        3.7
## 4 -0.7663125        4.3
## 5  1.4214450        4.4
## 6  0.5002196        4.2
```

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.
## -1.53900 -0.74460 -0.15640 -0.08835  0.45730  1.88200
```

Linear models

One model we can fit:

$y = \beta_0 + \epsilon$

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

## [1] 3.998272

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

##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    3.998      0.026 155.051        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 categorical predictors (the noise and degree data)
2. Simple linear models with continuous predictors.

For more sophisticated models, see the Linear Modeling notes. Next, we will discuss the last topic of this course, linear mixed models.