Introduction to R

Mandy Vogel

University Leipzig

November 10, 2015

Overview

Understand Hypothesis Tests

- Simulation Exercises

Recap

Machines

ergoStool

sleepstudy

Table of Contents I

Understand Hypothesis Tests
Simulation Exercises

Recap

Machines

ergoStool

sleepstudy

Z-test for a population mean

The z-test is a something like a t-test (it is like you would know almost everything about the perfect conditions. It uses the normal distribution as test statistic and is therefore a good example.

Objective

To investigate the significance of the difference between an assumed population mean μ_0 and a sample mean $\bar{x}$.

Limitations

1. It is necessary that the population variance σ^2 is known.
2. The test is accurate if the population is normally distributed. If the population is not normal, the test will still give an approximate guide.

Z-test for a population mean

Test statistic

$$Z = \frac{\bar{x} - \mu_0}{\sigma/\sqrt{n}}$$

1. Write a function which takes a vector, the population standard deviation and the population mean as arguments and which gives the Z score as result.
   - set a default value for the population mean
2. add a line to your function that allows you to process numeric vectors containing missing values!
3. the function `pnorm(Z)` gives the probability of $x \leq Z$. Change your function so that it has the p-value (for a two sided test) as result.
4. now let the result be a named vector containing the estimated difference, Z, p and the n.

You can always test your function using simulated values:

`rnorm(100,mean=0)` gives you a vector containing 100 normal distributed

Z-test for a population mean

Write a function which takes a vector, the population standard deviation and the population mean as arguments and which gives the Z score as result.

```
> ztest <- function(x,x.sd,mu=0){
+     sqrt(length(x)) * (mean(x)-mu)/x.sd
+ }
> set.seed(1)
> ztest(rnorm(100),x.sd = 1)
[1] 1.088874
```

Z-test for a population mean

Add a line to your function that allows you to also process numeric vectors containing missing values!

```
> ztest <- function(x,x.sd,mu=0){
+     x <- x[!is.na(x)]
+     if(length(x) < 3) stop("too few values in x")
+     sqrt(length(x)) * (mean(x)-mu)/x.sd
+ }
```

Z-test for a population mean

The function `pnorm(Z)` gives the probability of $x \leq Z$. Change your function so that it has the p-value (for a two sided test) as result.

```
> ztest <- function(x,x.sd,mu=0){
+     x <- x[!is.na(x)]
+     if(length(x) < 3) stop("too few values in x")
+     z <- sqrt(length(x)) * (mean(x)-mu)/x.sd
+     2*pnorm(-abs(z))
+ }
> set.seed(1)
> ztest(rnorm(100),x.sd = 1)
[1] 0.2762096
```

Z-test for a population mean

Now let the result be a named vector containing the estimated difference, Z, p and the n.

```
> ztest <- function(x,x.sd,mu=0){
+     x <- x[!is.na(x)]
+     if(length(x) < 3) stop("too few values in x")
+     est.diff <- mean(x)-mu
+     z <- sqrt(length(x)) * (est.diff)/x.sd
+     round(c(diff=est.diff,Z=z,pval=2*pnorm(-abs(z)),n=length(x)))
+ }
> set.seed(1)
> ztest(rnorm(100),x.sd = 1)
    diff         Z      pval         n
  0.1089    1.0889    0.2762  100.0000
```

Z-test for a population mean

Variants

1. Z-test for two population means (variances known and equal)
2. Z-test for two population means (variances known and unequal)

To investigate the statistical significance of the difference between an assumed population mean μ_0 and a sample mean $\bar{x}$. There is a function `z.test()` in the `BSDA` package.

Limitations (again)

1. It is necessary that the population variance σ^2 is known.
2. The test is accurate if the population is normally distributed. If the population is not normal, the test will still give an approximate guide.

Simulation Exercises

1. Now sample 100 values from a Normal distribution with mean 10 and standard deviation 2 and use a z-test to compare it against the population mean 10. What is the p-value?
2. Now do the sampling and the testing 1000 times, what would be the number of statistically significant results? Use `replicate()` (which is a wrapper of `tapply()`) or a `for()` loop! Record at least the p-values and the estimated differences! Use `table()` to count the p-vals below 0.05. What type of error do you associate with it? What is the smallest absolute difference with a p-value below 0.05?
3. Repeat the simulation above, change the sample size to 1000 in each of the 1000 samples! How many p-values below 0.05? What is now the smallest absolute difference with a p-value below 0.05?

Simulation Exercises – Solutions

- Now sample 100 values from a Normal distribution with mean 10 and standard deviation 2 and use a z-test to compare it against the population mean 10. What is the p-value? What the estimated difference?

```
> ztest(rnorm(100,mean=10,sd=2),x.sd=2,mu=10)["pval"]
  pval
0.0441
> ztest(rnorm(100,mean=10,sd=2),x.sd=2,mu=10)["diff"]
   diff
-0.0655
> ztest(rnorm(100,mean=10,sd=2),x.sd=2,mu=10)[c("pval","
  pval   diff
0.4515 0.1506
```

Simulation Exercises – Solutions

- Now do the sampling and the testing 1000 times, what would be the number of statistically significant results? Use `replicate()` (which is a wrapper of `tapply()`) or a `for()` loop. Record at least the p-values and the estimated differences! Transform the result into a data frame.

using `replicate()`

```
> res <- replicate(1000, ztest(rnorm(100,mean=10,sd=2),x.sd=2,mu=
> res <- as.data.frame(t(res))
> head(res)
     diff       Z   pval   n
1 -0.2834 -1.4170 0.1565 100
2  0.2540  1.2698 0.2042 100
3 -0.1915 -0.9576 0.3383 100
4  0.1462  0.7312 0.4646 100
5  0.1122  0.5612 0.5747 100
6 -0.0141 -0.0706 0.9437 100
```

Simulation Exercises – Solutions

- Now do the sampling and the testing 1000 times, what would be the number of statistically significant results? Use `replicate()` (which is a wrapper of `tapply()`) or a `for()` loop. Record at least the p-values and the estimated differences! Transform the result into a data frame.

using `replicate()` II

```
> res <- replicate(1000, ztest(rnorm(100,mean=10,sd=2),x.sd=2,mu=
+                          simplify = F)
> res <- as.data.frame(Reduce(rbind,res))
> head(res)
        diff       Z   pval   n
init -0.0175 -0.0874 0.9304 100
     -0.0751 -0.3757 0.7072 100
.1    0.0446  0.2232 0.8234 100
.2   -0.3642 -1.8209 0.0686 100
.3   -0.2039 -1.0195 0.3080 100
.4   -0.1872 -0.9359 0.3493 100
```

Simulation Exercises – Solutions

- Now do the sampling and the testing 1000 times, what would be the number of statistically significant results? Use `replicate()` (which is a wrapper of `tapply()`) or a `for()` loop. Record at least the p-values and the estimated differences! Transform the result into a data frame.

using `for()`

```
> res <- matrix(numeric(2000),ncol=2)
> for(i in seq.int(1000)){
+     res[i,] <- ztest(rnorm(100,mean=10,sd=2),x.sd=2,mu=10)[c("pval","diff")] }
> res <- as.data.frame(res)
> names(res) <- c("pval","diff")
> head(res)
    pval    diff
1 0.0591 -0.3775
2 0.2466  0.2317
3 0.6368  0.0944
4 0.5538 -0.1184
5 0.9897 -0.0026
6 0.7748  0.0572
```

Simulation Exercises – Solutions

- Use `table()` to count the p-vals below 0.05. What type of error do you associate with it? What is the smallest absolute difference with a p-value below 0.05?

```
> table(res$pval < 0.05)

FALSE  TRUE
  960    40
> tapply(abs(res$diff),res$pval < 0.05,summary)
$`FALSE`
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0002  0.0585  0.1280  0.1411  0.2068  0.3847

$`TRUE`
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.3928  0.4247  0.4408  0.4694  0.5102  0.6859

> min(abs(res$diff[res$pval<0.05]))
[1] 0.3928
```

Simulation Exercises – Solutions

- Repeat the simulation above, change the sample size to 1000 in each of the 1000 samples! How many p-values below 0.05? What is now the smallest absolute difference with a p-value below 0.05?

```
> res2 <- replicate(1000, ztest(rnorm(1000,mean=10,sd=2),
+                                x.sd=2,mu=10))
> res2 <- as.data.frame(t(res2))
> head(res2)
     diff       Z   pval    n
1 -0.0731 -1.1559 0.2477 1000
2  0.0018  0.0292 0.9767 1000
3  0.0072  0.1144 0.9089 1000
4 -0.1145 -1.8100 0.0703 1000
5 -0.1719 -2.7183 0.0066 1000
6  0.0880  1.3916 0.1640 1000
```

Simulation Exercises – Solutions

- Repeat the simulation above, change the sample size to 1000 in each of the 1000 samples! How many p-values below 0.05? What is now the smallest absolute difference with a p-value below 0.05?

```
> table(res2$pval < 0.05)

FALSE  TRUE
  946    54
> tapply(abs(res2$diff),res$pval < 0.05,summary)
$`FALSE`
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00010 0.02092 0.04285 0.05149 0.07400 0.22610

$`TRUE`
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00240 0.02115 0.04535 0.05435 0.08433 0.14760
```

Simulation Exercises Part II

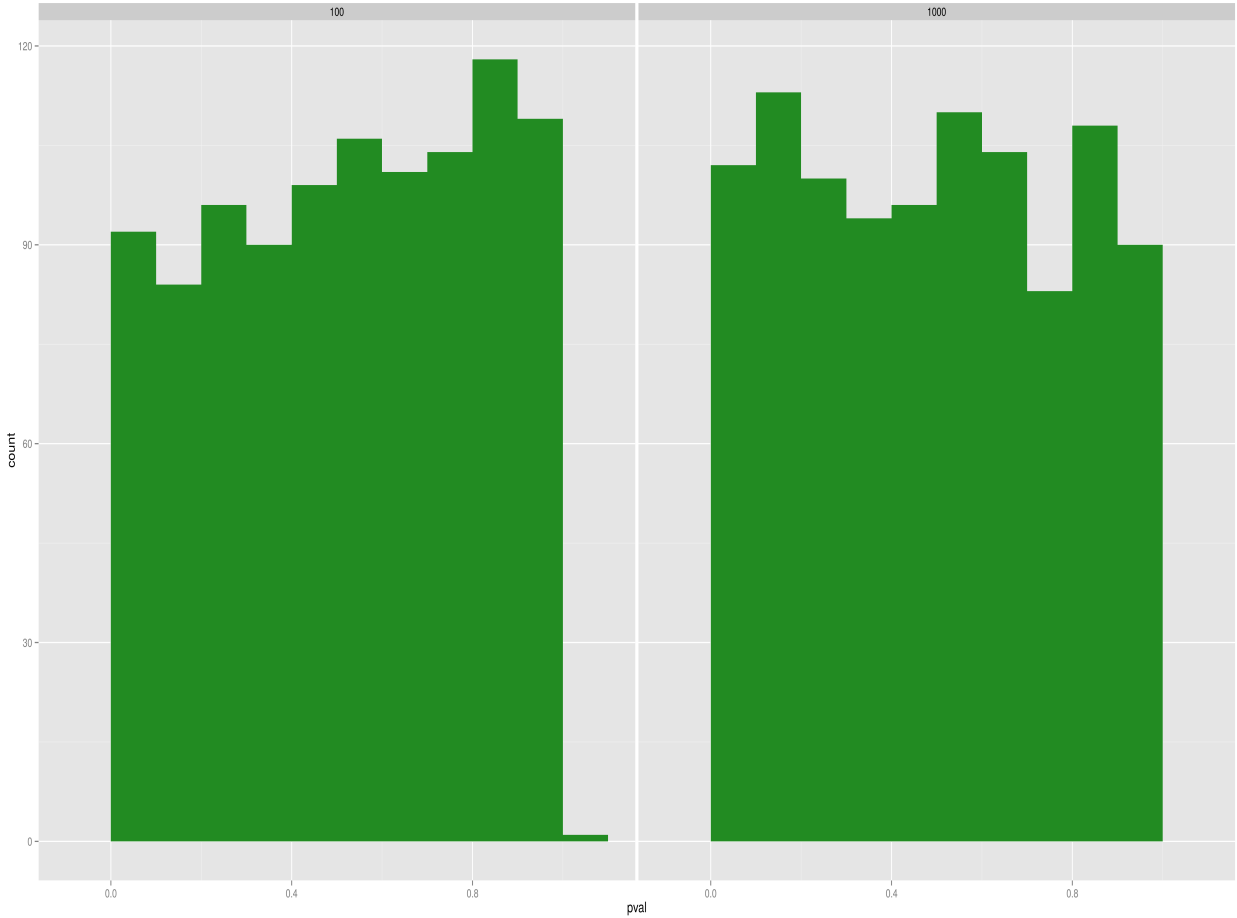
1. Concatenate the both resulting data frames from above using `rbind()`
2. Plot the distributions of the pvals and the difference per sample size. Use `ggplot2` with an appropriate geom (density/histogram)
3. What is the message?

Simulation Exercises – Solutions

- Concatenate the both resulting data frames from above using `rbind()`
- Plot the distributions of the pvals and the difference per sample size. Use `ggplot2` with an appropriate geom (density/histogram)

```
> res <- rbind(res,res2)
> require(ggplot2)
> ggplot(res,aes(x=pval)) +
+     geom_histogram(bin=0.1,fill="forestgreen") +
+     facet_grid(~ n)
> ggsave("hist.png")
```

Simulation Exercises – Solutions

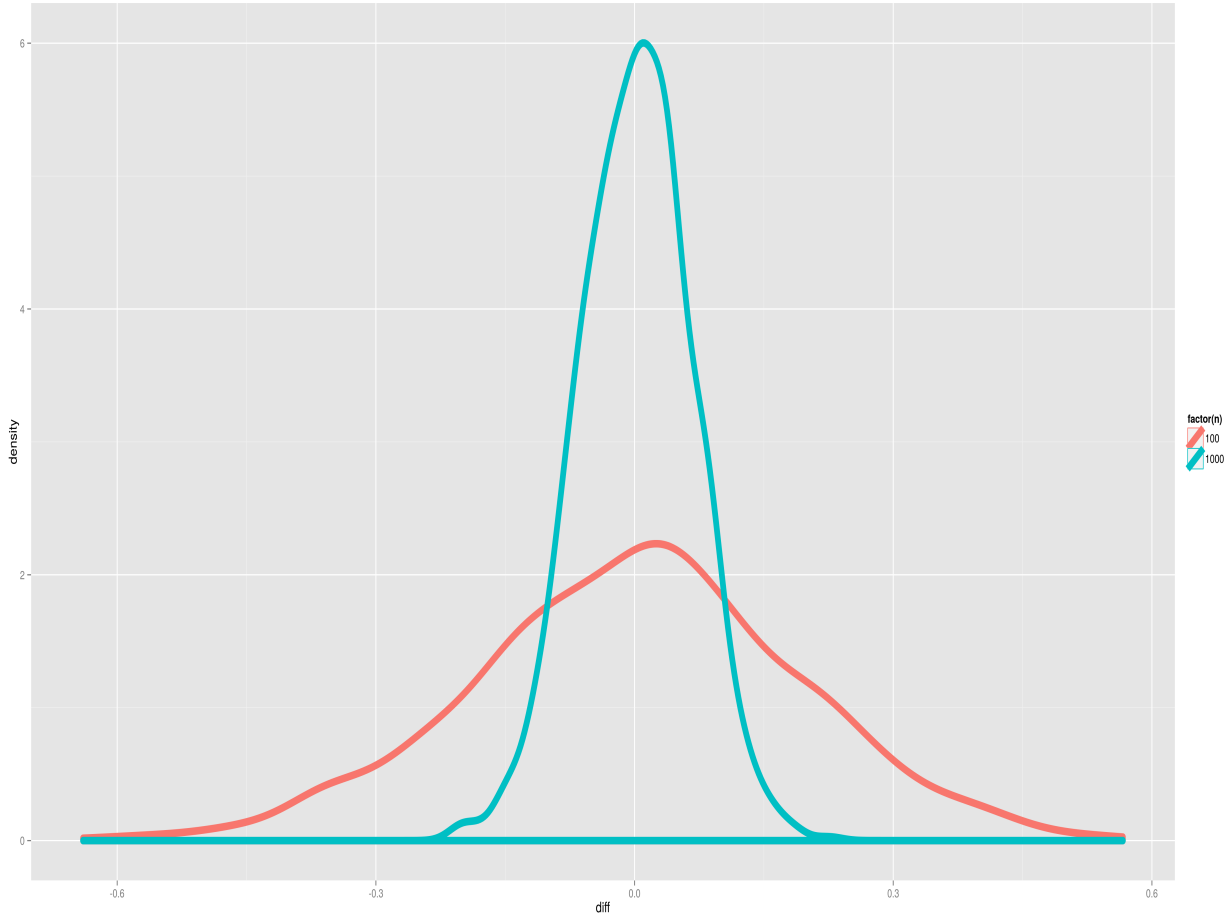


Simulation Exercises – Solutions

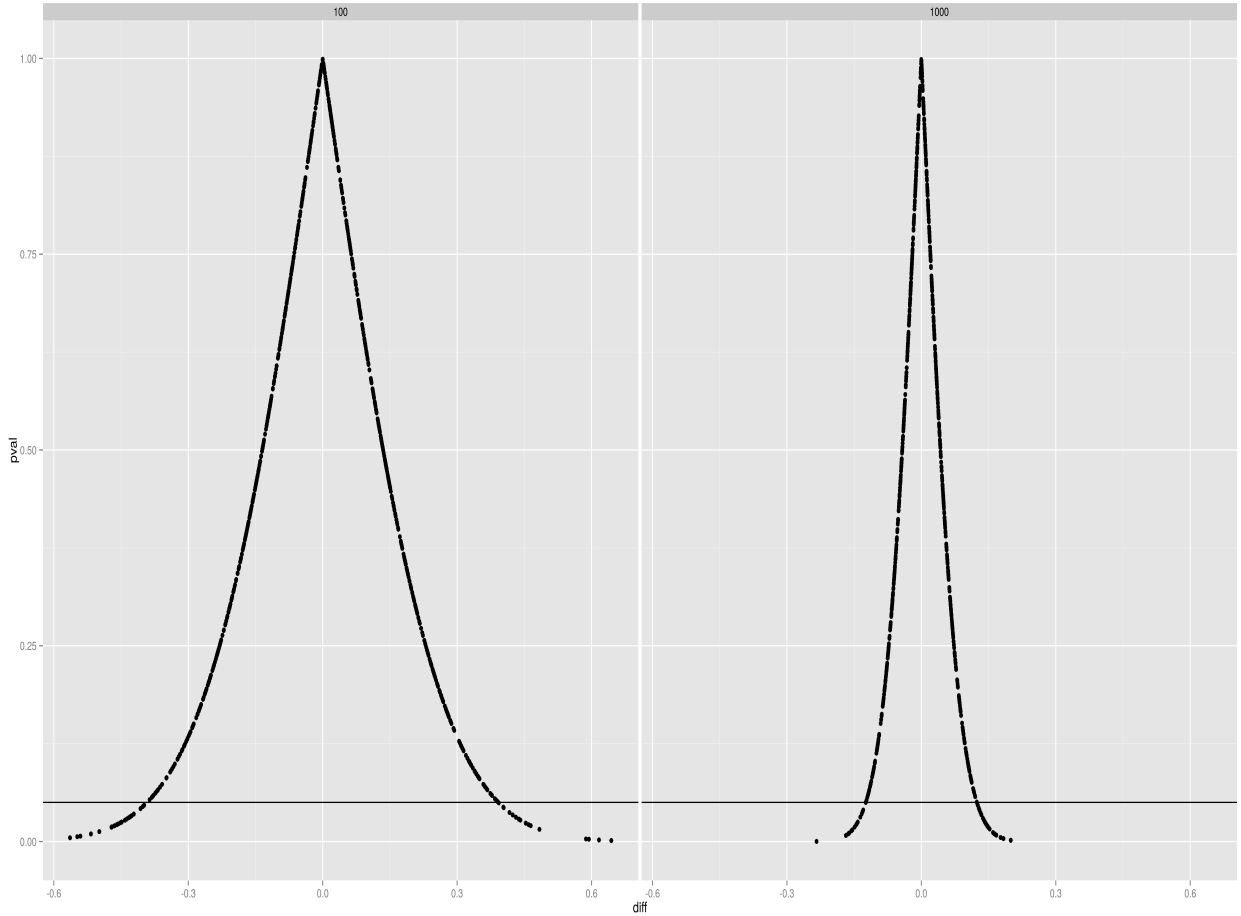
- Plot the distributions of the pvals and the difference per sample size. Use `ggplot2` with an appropriate geom (density/histogram)

```
> ggplot(res,aes(x=diff,colour=factor(n))) +
+     geom_density(size=3)
> ggsave("dens.png")
```

Simulation Exercises – Solutions



Simulation Exercises – Solutions



Simulation Exercises – Solutions

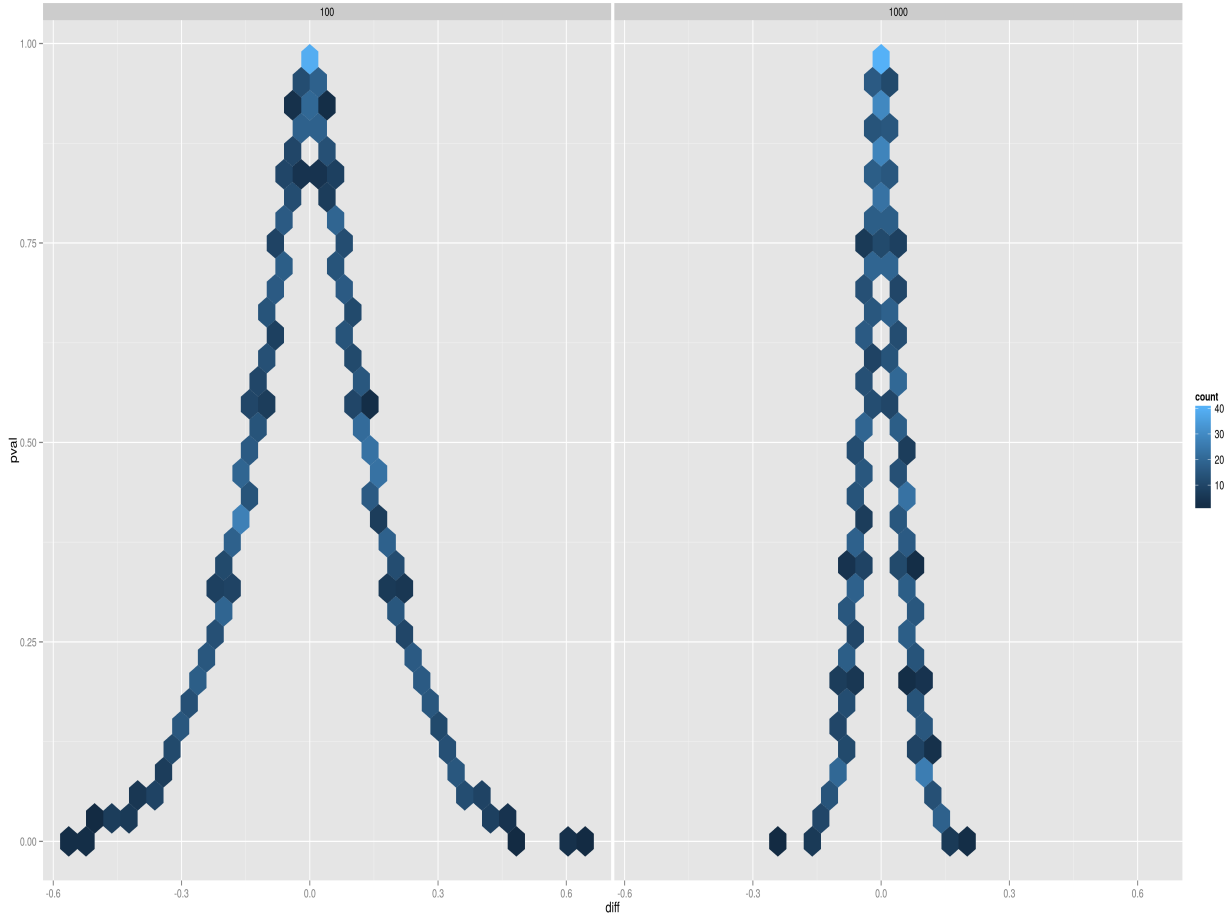


Table of Contents I

Understand Hypothesis Tests
Simulation Exercises

Recap

Machines

ergoStool

sleepstudy

Linear Mixed Models

- mixed models incorporate fixed-effect parameters and random effects
- two of the main packages for fitting mixed models: `lme4` and `nlme`
- in `lme4`: the general syntax is `lmer(formula, syntax)`; random effects are enclosed in brackets, grouping factors are indicated by a pipe symbol
- fixed effects are parameters, random effects are unobserved random variables

Table of Contents I

Understand Hypothesis Tests
Simulation Exercises

Recap

Machines

ergoStool

sleepstudy

Building Linear Mixed Models

- “Statistical model building is still somewhat more of an art than a science...” (Douglas Bates)
- there are many practical issues that we should bear in mind
- interpretation of cœfficients the model’s structure can be subtle

Categorical Covariates with Fixed Effects - data

- data:
  - `Machines` data (`MEMSS` package)
  - measure productivity on a manufacturing task according to the type of machine used and the operator

Categorical Covariates with Fixed Effects - data

1. load the data (i.e. install/load the resp. package if necessary)
2. look at the data structure (which command?)
3. is the design balanced or unbalanced?
4. if one covariate is fixed and one random, which would you choose for each category and why?
5. try to visualize the data appropriately. What can you get out of the visualization?

Models

1. fit the following models!

```
> m.malm <- lm(score ~ Machine, data = Machines)
> m.ma1 <- lmer(score ~ Machine + (1|Worker), data = Machines)
> m.ma2 <- lmer(score ~ Machine + (Machine|Worker),
+     data = Machines)
> m.ma3 <- lmer(score ~ Machine +
+                   (1|Worker) + (1|Machine:Worker),
+                   data = Machines)
```

2. which is the most complex one?
3. examine the fixed effects in the models. Compare them!
4. load the `lmerTest` package, refit the models, examine the fixed effects including p-values.
5. now use `anova()` to compare the three models. Which model seems to be the most appropriate?

Table of Contents I

Understand Hypothesis Tests
Simulation Exercises

Recap

Machines

ergoStool

sleepstudy

Random vs Fixed effect - data

- data:
  - `ergoStool` data (`MEMSS` package)
  - measurement of effort to arise from a stool

Random vs Fixed effect - data

1. fit the following models

```
> m.es1 <- lmer(effort ~ 1 + (1|Subject) + (1|Type),
+               data = ergoStool,
+               REML=FALSE)
> m.es2 <- lmer(effort ~ 1 + (1|Subject),
+               data = ergoStool,
+               REML=FALSE)
> m.es3 <- lmer(effort ~ 1 + Type + (1|Subject),
+               data = ergoStool,
+               REML=FALSE)
> m.es4 <- aov(effort ~ 1 + Type + Subject,
+              data = ergoStool)
```

2. what are the interpretations of each of them? which parameters are estimated?
3. for which of them makes the `glht()` function sense? why?

Table of Contents I

Understand Hypothesis Tests
Simulation Exercises

Recap

Machines

ergoStool

sleepstudy

LMM for longitudinal - data

- data:
  - `sleepstudy` data (`MEMSS` package)
  - effect of sleep deprivation on reaction time

LMM for longitudinal

1. use ggplot to visualize the reaction times dependent on time
2. add the regression line for a simple linear regression per subject
3. is the design balance or unbalanced?
4. fit the following models? correct errors

```
> m.ss1 <- lmer(Reaction ~ 1 + Days + (1 + Days|Subject),
+                 data = sleepstudy)
> m.ss2 <- lmer(Reaction ~ 1 + Days +
+               (1|Subject) + (Days|Subject), sleepstudy)
```

5. which model incorporates more estimates (i.e. less degrees of freedom left, if you use lmerTest); use the output of `summary()` and an `anova()` comparison of the two models
6. which of them would you prefer and why?

Understand the following lines I

```
> require(broom)
> coef.mm <- as.data.frame(coef(m.ss2)[["Subject"]])
> coef.mm$model <- "mixed"
> coef.mm$Subject <- row.names(coef.mm)
>
> sleepstudy.l <- split(sleepstudy,sleepstudy$Subject)
> tmp <- lapply(sleepstudy.l, function(x){
+     m <- lm(Reaction ~ Days, data = x)
+     data.frame(t(coef(m)))
+ })
> coef.sm <- Reduce(rbind,tmp)
> coef.sm$model <- "simple lm"
> coef.sm$Subject <- names(tmp)
>
> names(coef.mm)[1:2] <- names(coef.sm)[1:2] <- c("intercept","sl
> coefs <- rbind(coef.mm,coef.sm)
> require(grid)
```

Understand the following lines II

```
> ggplot(coefs,aes(x = intercept, y = slope, shape = model)) +
+   geom_point(aes(colour = model), size = 5) +
+   geom_path(aes(group = Subject), arrow = arrow(ends = "first"))
+   annotate(geom = "point",
+            x = fixef(m.ss2)[1],y = fixef(m.ss2)[2],
+            size = 6, colour = "darkgreen")
```