Experimental Design and Data Analysis Course VU

Tim de Boer

February-March 2021

1 Lecture 1

Contents: recap of statistical concepts.

- The normal density curse is given by:

$$f_{\mu,\sigma}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp^{-\frac{1}{2}(x-\mu)^2/\sigma^2}$$

  Where μ determines the position of the peak on the x-axis and σ determines the width of density curve.

- A quantile of α is number q_α such that $P(X \leq q_\alpha) = \alpha$. The upper quantile is the other side, $P(X > q_\alpha) = \alpha$. The quantile value q_α is the median value if we choose $\alpha = 0.5$. If $\alpha = 0.25$, q_α is the value such that the data is split in 25 below and 75% above q_α. The function qnorm() aims to find the boundary value, A in P($X < A$), given the probability P. For example, suppose you want to find the 85th percentile of a normal distribution whose mean is 70 and whose standard deviation is 3. Then you ask for:

```
qnorm(0.85,mean=70,sd=3)
```

- A QQ-plot can reveal whether data follows certain distribution P. It plots the theoretical ordered probabilities from normal distribution on the x-axis (theoretical quantiles) versus the from sampling obtained quantiles on y-axis. Linear line means correct distribution (sampled from the population). Use the qqnorm plot together with a histogram to see if the data is normally distributed. Also plot an boxplot to get an idea about differences and spread.

```
par(mfrow=c(1,2)); qqnorm(data); hist(data); boxplot(hours~environment)
```

- Central Limit Theorem: Sampling from a (not known) distribution and calculating mean for these sample means. The distribution of all these sample means is more normally distributed. If you keep sampling from the unknown distribution and keep calculating the mean of these samples, the distribution of means becomes more and more normally distributed. The higher sample size, the better normally distributed. When a sample is taken from the distribution N(μ, σ^2) then the sample mean is N($\mu, \sigma^2/n$): another way of describing the Central Limit Theorem: the sample mean varies less than original mean.

- In a real dataset, the full population std σ is unknown. We replace σ with sample std we call s which gives the T-distribution as a sort of Central Limit Theorem if we take the mean of samples:

$$T = \frac{\bar{X} - \mu}{s/\sqrt{n}}$$

  which does not have N(0,1) distribution due to uncertainty about full population. Instead, T has t-distribution with n - 1 degrees of freedom.

- A point estimate for a unknown parameter (for example the mean) is a function of *only* the observed data, seen as a random variable. Denote them with $\hat{\mu}$.
- The confidence interval of $1-\alpha$, e.g. 95%, is a random interval based only on the observed data that contains the true value of the parameters with probability of 95%. If σ is unknown (which is true in almost all cases), the t-confidence interval becomes $[\bar{X}-t, \bar{X}+t]$ e.g. how confident are we that that true proportion is is in between 2 std's from the sample proportion $\hat{p}$. If we want to calculate a 95% confidence interval for a normal distributed ppopulation, we have to calculate the 97.5th percentile:

$$CI_{range} = \mu \pm qnorm(97.5) \cdot \frac{\sigma}{\sqrt{n}}$$

  And for a sample of the population we use the t-distribution:

$$CI_{range} = \mu \pm qt(97.5, n-1) \cdot \frac{s}{\sqrt{n}}$$

  With s the standard deviation of the sample instead of σ which we use for the whole population. In R for normal distributed population:

```
mu = mean(birthweights); sd = sd(birthweights); size = length(birthweights)
error = qnorm(0.975)*sd / sqrt(size) # or with qt if we have sample
lowerbound = mu - error; upperbound = mu + error
```

- Strong outcome: H_0 rejected, H_1 is true. Weak outcome: H_0 not rejected. Type 1 error: rejecting H_0 while it is true, type 2: not rejecting H_0 while it is false.
- Power depends on amount of data: 1-Probability(type 2 error), thus power is the probability of correctly rejecting H_0 (seeing an effect which is really an effect). If we want to know the power of our test, we repeat the test 1000 times where we initialize the distribution of our sample x and y based on parameters, do a t-test, and then calculate how often the p value is below our threshold of 0.05. We can calculate this fraction as the mean of the total amount of tests. For this example, the null hypothesis we are testing is $H_0 : nu = mu$.

```
b = 1000; nu = 175, mu = 180, m = n = 30; sd=5; p_values = numeric(b);
for (b in 1:B) {
    x=rnorm(n,mu,sd); y = rnorm(m,nu,sd);
    p[b] = t.test(x,y,var.equal=TRUE)[[3]]} #3rd value is the p-value for our H_0
    power= mean(p<0.05)}
```

- There are three ways to reject H_0: t-value bigger than quantile, p-value lower than 0.05, mean not in confidence interval (so the mean we want to test is not in the range of the calculated mean of the sample plus or minus 2 std).
- Since we don't know the distribution, we generally use the t-distribution with $t_{0.025,n-1}$ (2.5%, n-1 degrees of freedom); this makes CI bigger (more conservative) since $t > z$, which was 1.96.
- Two sample t-test we calculate by subtracting sample means and dividing by standard error of the two samples (e.g., adding SE1 with SE2, divide by Size1 + size2 - 2) from which we have our T. Unreliable for Size below 20. In R it is simple: $t.test(x, y$, and create x an y with $x = rnorm(size, mean, variance)$.
- For one-sample test (is the data mean equal / smaller / bigger to / than a certain mean?) we can use t-test or sign-test. For normal data, t-test has bigger power (closer to 1), since t-test has a stronger assumption (data must be normal) and thus better performance than sign-test for normal data, since sign-test does not assume a normal distribution. We can do a one-sided t-test as follows, in this case to check if mean is bigger than 2800:

```
t.test(birthweights, mu = 2800, alternative = "greater", conf.level = 1 - alpha)
```

2 Lecture 2

Contents: bootstrap (sort of simulations) confidence intervals and tests, one sample tests (t-test, sign test, Wilcoxon signed rank test).

- **Bootstrap Confidence Interval:**
  - With bootstrapping, we sample a random point from our data, put the point back in our data, and sample again etc. For each bootstrap sample, we estimate a statistic T, for example the mean or median.
  - We do this bootstrapping for B times, where B is mostly around 10.000.
  - The power of your data does not increase with B, but the confidence of your bootstrap estimate does.
  - Then use the formula for CI $[2 * T - Tq_{975}, 2 * T - Tq_{25}]$ where Tq_{975} is the area of distribution curve higher than 97.5% and Tq_{25} for lower than 2.5%. Then you know in which interval the true mean or median exists.

```
bootstrap_interval = function(data){
    T = median(data)
    results = numeric(1000)
    for (i in 1:1000){
    surrogate_data = sample(data, replace=TRUE)
    results[i] = median(surrogate_data)}
    median25 = quantile(results,0.025); median975 = quantile(results,0.975)
    return(c(2*T - median975,2*T - median25))}
bootstrap_interval(data)
```

  - Another way to calculate a bootstrap confidence interval is by using the Central Limit Theorem: we assume our data comes from a certain distribution, but the sample mean will always have a distribution of approximately $N(\mu, \sigma^2/n)$. So if we sample our data we can use the formula for the CI:

$$CI_{range} = qnorm(0.975) \cdot \frac{s}{\sqrt{n}}$$
$$upperbound = \mu + CI_{range}; lowerbound = \mu - CI_{range}$$

  So, when we think our data comes from a $Exp(\lambda)$ distribution, we first calculate the CI of the sample; then use this to calculate the CI of the λ's by using

$$\lambda = \frac{1}{mean(Exp(\lambda))}$$

  From which we can calculate the CI of the median with:

$$median(Exp(\lambda)) = \frac{ln2}{\lambda}$$

  In R:

```
error = qnorm(0.975) * (sd(data)/sqrt(length(data)))
upperbound = mean(data)+error; lowerbound =  mean(data)-error);
lower_lambda = 1/upperbound; upper_lambda = 1/lowerbound
# dividing so upperbound gives lower lambda
lower_median = ln(2)/upper_lambda; upper_median = ln(2)/lower_lambda
# again dividing so upper_lambda gives lower_median
```

- **Bootstrap tests:**
  - For tests, we sort of generate new data (whereas with Bootstrap Confidence Interval we sample from same dataset).
  - Suppose we have a sample, a H_0 and test statistic T, but we lack distribution of T under H_0. Now we do not have critical value for T so we cannot perform test.
  - We can now simulate Pseudo Observations characterizing H_0 with bootstrap test which uses simulations.
  - We do B simulations, we generate fake data (same sample size as original) according to our H_0, we compute T, we compare this T with the original T to determine p-value.
  - For example, calculate the max of our original data, and calculate the max of each bootstrap sample. We make a histogram from that, and then we plot the max of our original data in this histogram and we calculate the area to the smallest side of that (if this area is lower than 0.05, then H_0 rejected: the means are different between H_0 and our data).
  - For an example in R, we make surrogate data based on our H_0 hypothesis: our original data comes from distribution of Exp(λ). We test this by calculating the test statistic Median for the surrogate data for 1000 times. We plot our test statistic of our original data in the histogram of the test statistics of the 1000 surrogate datasets and calculate the smallest area, e.g. the area left or right from original median; this is the probability that the data stems from our H_0 distribution:

```
bootstrap_test_median = function(data, gamma){
    t = median(data); # Save t of sample
    results = numeric(1000);
    for (i in 1:1000){
    surrogate_data = rexp(length(data),rate=gamma) # Exponential sample
    results[i] = median(surrogate_data)} # t* of sample
    # Calcuate p of both lower and higher tail, take the smallest.
    p = min(sum(results<t)/1000,sum(results>t)/1000); return(p) }
```

- **Normal tests:** one sample t-test, see explanaion in the notes of lecture 1. Recall the code in R:

```
t.test(birthweights, mu = 2800, alternative = "greater", conf.level = 1 - alpha)
```

- **Non-normal tests:**
  - the sign test is very simple, if you think the median of your data is x, you calculate how many data point are above or below that; this should be approximately half of the data. Then we do a binomality test to get test statistics (since if more than half of our data is above or below median, this does not mean H_0 could be rejected; could not be significant). In R:

```
amount = sum(data >= 40)
binom.test(amount, length(data), p=0.5, alternative="less")[[3]]
```

  - Wilcoxon test assumes a symmetric population (a stronger assumption than sign test has). Test statistics is as follows: we substract our suspected median from all data points, calculate the rank of all datapoints, then sum the rank of the datapoints for which the value is bigger than the suspected median. Large sum indicates true median is higher than suspected median. Small sum indicates true median is lower. If you use this test in R you get a corresponding p-value.

```
wilcox.test(data,mu>=40)
```

3 Lecture 3

Contents: two-paired samples and two independent samples test

- **Two paired samples**
  - Paired t-test:
    - The differences between samples assumed to be normal. H_0 is that mean of differences is 0. Test statistic is:
      $$T = \frac{Z_N}{S_N}$$
      Z is average of differences and S is sample standard deviation.
    - In R, paired=True for t.test is given to tell samples are paired (e.g., same person).
    - We could also calculate the differences ourselves and use one sample t-test.
    - T-test assumes normal distribution: use qq plot or shapiro.test (p high means no reason to suspect differences are not from normal population).

```
data_softdrink = subset(data, data$drink == 'lemo')
data_energydrink = subset(data, data$drink == 'energy')
t.test(data_softdrink$before, data_softdrink$after, paired = T)
t.test(data_energydrink$before, data_energydrink$after, paired = T)
```

  - Pearson Correlation:
    - Assumes normality.
    - Test calculates the correlation p: cor(X,Y). Equation:
      $$p = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum(X_i - \bar{X})^2 \sum(Y_i - \bar{Y})^2}}$$
      From which we can calculate a test statistic with n-2 degrees of freedom.
    - In R, with cor.test, Pearson is used as default. Low p-value is significant correlation.
  - Spearman's rank correlation:
    - This test does not assume normality.
    - Compares ordering of ranks of X and Y. If the data are rank correlation, these sequences will run in parallel or opposite.
    - In R, with cor.test with method="spearman".

```
cor.test(data$before, data$after, method = "pearson")
cor.test(data$before, data$after, method = "spearman")
# could also make a scatter plot with correlation included:
library("ggpubr")
ggscatter(data, x = "before", y= "after", add = "reg.line", conf.int = TRUE,
          cor.coef = TRUE, cor.method = "pearson",
          xlab = "Before drink runtime [sec]",
          ylab = "After drink runtime [sec]",
          title = "Scatter plot runtime")
```

  - Permutation test:
    - Do not assume normality.
    - Can use any test-statistic.
    - Like in bootstrap-test, we simulate distribution of T under H_0 (repeating B times).

* We generate new values by generating a permutation of the original data (relabeling), e.g. choose between (X,Y) and (Y,X) with equal probability.
* We assume that permutating the distribution of X and Y within pairs would not change the value of the test-statistic.
* We plot a histogram of the outcome of the test statistic of all the 1000 permutations; then we plot our value of test statistic in this histogram to see our p-value. Low p-value means that our data does not come from the H_0 distribution, in other terms, there is a difference.
* The most common test-statistic is mean of differences X-Y within pairs.

```
test_statistic = function(x,y) {mean(x-y)}
B = 1000; tstar = numeric(B)
for (i in 1:B){
  runtime_star = t(apply(cbind(data_softdrink$before, data_softdrink$after),
  1, sample))
  tstar[i] = test_statistic(runtime_star[,1], runtime_star[,2])
}
myt = test_statistic(data_softdrink$before, data_softdrink$after)
hist(tstar, main = 'Histogram tstar softdrink')
pl = sum(tstar<myt)/B
pr = sum(tstar>myt)/B
p_value = 2*min(pl,pr) # multiply by 2 to be sure p-value is significant
```

- Two independent samples
  - Two samples t-test:
    * Assumes both samples from normal population.
    * Test that means of population are the same as null.
    * t-test with two argument performs by default for independent samples.
  - Mann-Whitney test:
    * Based on ranks with test statistic median with H_0 of equal median between X and Y.
    * Considers M+N ranks in the combined sample of X and Y of length M+N.
    * If population are the same, M rank should lie randomly between 1 and M+N.
    * We calculate sum of ranks from X. If this value is large, this indicates that X-values are bigger than Y-values.

```
# calculate differences to be able to compare soft with
# energydrink (independent test), instead of before and after
# the run comparison (which was a paired test)
data_softdrink['differences'] = data_softdrink$after - data_softdrink$before
data_energydrink['differences'] = data_energydrink$after - data_energydrink$before
t.test(data_softdrink$differences, data_energydrink$differences, paired = F)
```

  - Kolmogorow-Smirnov test:
    * Based on the differences in the histograms of the samples X and Y.
    * The test statistic computes the maximal vertical differences in the summed histograms (empirical distribution functions), see figure 1.
    * A large difference in summed histogram indicate that the means are different.

```
ks.test(light1,light2)
```

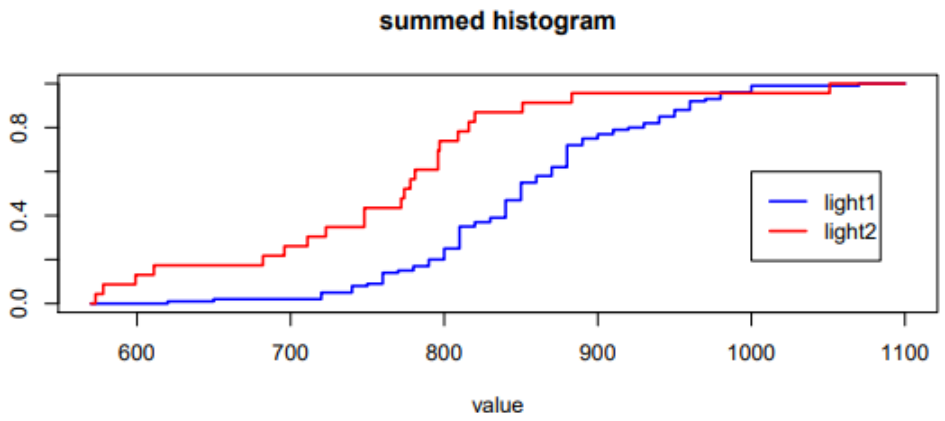

Figure 1: Summed histograms of two samples. The y-axis is the ratio of instances with a certain value. The blue line has higher mean, e.g. more of the instances have higher values.

4 Lecture 4

Contents: one-way ANOVA, Kruskal-Wallis and permutation tests in setting of one-way ANOVA.

- **One-way ANOVA:**
  - Multiple groups (also called levels, or categories).
  - Using equal number of units (sample size), balanced design, is preferable.
  - Anova assumes data from normal distributions.
  - With two groups, the setting is same as two sample t-test.
  - This test is as default setting with the first group as reference class.
  - The equation of test statistic F is:

$$F = \frac{betweengroupSS}{withingroupSS} = \frac{meanSS_A}{meanRSS} = \frac{\sum n_i(\bar{Y}_i - \bar{Y})^2/(I-1)}{\sum_I \sum_{n_i}(Y_{ij} - \bar{Y}_i)^2/(n-I)}$$

    $\bar{Y}_i$ is mean of a group and $\bar{Y}$ is mean of all groups, so between groups SS. Below is withingroups, divided by sample size minus amount of groups. See figure 2 for all output.

Source	Df	Sum Sq	Mean Sq	F value	p-value
Factor A	$I-1$	SS_A	$SS_A/(I-1)$	$f = \frac{SS_A/(I-1)}{RSS/(n-I)}$	$P(F > f)$
Residuals	$n-I$	RSS	$RSS/(n-I)$		
Total	$n-1$	SS_T			

Figure 2: Table contents of one-way ANOVA output.

  - Large F give more evidence against H_0.
  - Input of Anova of R is two columns, one column containing the data values, second column containing the group to which the data belongs.
  - The summary of the output of the ANOVA depends on which contrast you used (see above about the first group as reference class). So other means can be calculated as $\mu_i - mu_1$ since only the differences with first group are in the output.
  - Plot a qqplot with qqnorm(residuals(anova output)) which plots a qqplot of data of all groups combined to get a better estimate of normality of all data combined and corrected for the fact that they are sampled from other distributions.

– If no normality but you still want to use ANOVA, you could try to transform data (with log transform for example, omitting outliers (but be careful with that))

```
# data should be two columns, one with outcome y and
# one with treatment levels as factors
# we could transform data with for example:
ratframe=data.frame(worms=as.vector(as.matrix(ratdata)), group=rep(1:4,each=5))
attach(ratframe)
as.factor(group)
rataov=lm(worms~group)
anova(rataov)
summary(rataov)
#check normality
qqnorm(residuals(rataov)); #linear line inidicates normality
plot(fitted(rataov),residuals(rataov))
#if spread doesn't differ over x-axis, we have equal variance
detach(ratframe)
```

- **Kruskal-Wallis test:**
  - A non-parametric counterpart of ANOVA which does not rely on normality but on ranks (thereby a bit less powerful results than 1-way ANOVA).
  - Setting and design same as 1-way ANOVA.
  - Computes the sum of ranks of each group within the total data (same as Mann-Whitney!).
  - Does assume sample size bigger than 5 per group.

```
attach(ratframe)
kruskal.test(worms, group)
detach(ratframe)
```

- **Permutation tests:**
  - We calculate the sum of residuals of original data and compare this with the average of the B (1000) simulations of random samples from our original data with permutation of the labels.
  - If sum of residuals of our original data has low p-value in the histogram of the simulations, we know that there is significant difference, e.g. the difference is not by chance.
  - Permutation test independent samples: Permute the labels and attach then to data

```
dogs=read.table("dogs.txt",header=TRUE)
treat=factor(rep(1:3,c(10,10,10)),labels=c("iso","halo","cyclo"))
dogsdata=data.frame(plasma=as.vector(as.matrix(dogs)),treat)
boxplot(plasma~treat,data=dogsdata)
mystat=function(x) sum(residuals(x)^2)
B=1000
tstar=numeric(B)
for (i in 1:B) {
  treatstar=sample(treat) ## permuting the labels
  tstar[i]=mystat(lm(plasma~treatstar)) }
myt=mystat(lm(plasma~treat))
hist(tstar)
pl=sum(tstar<myt)/B
pr=sum(tstar>myt)/B
p_value = 2*min(pl,pr)
```

5 Lecture 5

Contents: two-way ANOVA, general factorial design, randomized block design, repeated measures.

- Two-Way ANOVA works as One-way ANOVA, only now we have multiple factors with possibly multiple levels. We first investigate the interaction effect of the two factors. If interaction effect is non-significant, we are able to investigate the effects of the factors individually. We now do factor 1 PLUS factor 2 instead of the multiplication sign. Now we can look at the p-values for these factors. We need more than 1 observation per cell for R to calculate something! Of course also check normality and equal variance by the residuals-qqnorm and the fitted plot.

```
model = lm(y\backsim f1*f2, data=data)
anova(model)
#non-significant interaction effect? Only then check additive model
model2 = lm(y\backsim f1+f2, data=data)
anova(model2)
#check assumptions
qqnorm(residuals(model2)
plot(fitted(qqnorm),residuals(qqnorm))
```

- Additional insight is achieved with an interaction plot, which fixes one factor and plots the average outcome (vertical axis) against the levels of the other factor (horizontal axis). Interaction shows up as nonparallel curves.

```
attach(data)
interaction.plot(environment,humidity,hours);
interaction.plot(humidity,environment,hours)
```

- With Randomized Block Design, we suppose we know the effect of one factor which we call 'block'. Then we can create homogeneous groups of experimental units, in which the treatment effect is easier to see and not blurred by variation due to the block factor. We still use Two-way Anova to test but now we are just interested in the factor which is not the block factor. The advantage of the block design is that more precise conclusions can be obtained by removing variation, present due to block factor. The units must be similar within the blocks, and dissimilar between the blocks. An example is with a repeated measures test, where we presume beforehand that subject person (id) will have significant effect on our outcome and thus we assign this as our block factor. Taking repeated measures is attractive, because fewer experimental units are needed and "extraneous" variation between units is reduced. However, in many studies, in particular most "longitudinal studies", where individuals are followed over time, the assumption of "exchangeability" fails. More complicated models are then necessary. Also think about learning effects.

```
datalm=lm(pain~factor+block,data=data);
anova(datalm)
```

6 Lecture 6

Contents: 1 Friedman test 2 incomplete block design 3 random effects 4 crossover design 5 split-plot design 6 overview block designs

- **Friedman test** is a Two-way Anova for data which is non-normally distributed. For example we take a repeated measures test, for which we can use the Friedman test, testing the relevance of one factor (e.g., drug), taking into account the blocking factors (e.g., id). If significant, H_0 is rejected: there is a treatment effect. We can compare the results with ANOVA which also gives significant effect for drug factor (and also for id but that is expected from a block factor, e.g. every person is different).

```
friedman.test(duration,drug,id,data=itchdata)
itchaov=lm(duration $\sim$ drug+id);
anova(itchaov)
```

- **Incomplete block design**: if every treatment is applied to every block this may lead to a lot of experiment. We use incomplete block design to select subset of experiments to execute in a balanced way. The outcome is measured (only) for the combinations marked by a *, 30 experiments in total, 3 per block. Every pair of treatments is compared within exactly 2 blocks, see figure 3. The analysis is the same as for an ordinary block design.

	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10
A		*		*			*	*		*
B	*	*				*	*		*	
C	*	*	*	*	*					
D			*		*		*		*	*
E				*	*	*		*	*	
F	*		*			*		*		*

Figure 3: Example incomplete block design with treatment factor A to F and block factor (example: id) 1 to 10

- **Random effects:** Don't choose the blocks as fixed, but as a random selection of all possible blocks (the block population, e.g. don't choose the same 5 professors to grade exam, choose randomly 5 out of 10 professors of the population). Treatment factor is fixed (exam 1 and exam 2). So, we are not interested in the professors (random block effect), but in the exam difficulty we are (fixed treatment effect). This is a combination of random and fixed effects in our design, thus a *mixed effects model.*
- **Cross-over design:** Take a random sample of experimental units from the relevant population. Divide the units at random in two equal groups. Apply the treatments in one order to the units in the first group, and in the reversed order to the units in the second. An order effect of the outcomes is suspected. Example: Comparing pain relief by a dedicated drug or by a placebo. Both treatments are applied to every individual (with recovery time in between), see figure 4. Fixed effects:

		period 1	period 2
sequence	$T_1\,T_2$	μ	$\mu+\alpha+\beta$
	$T_2\,T_1$	$\mu+\alpha+\gamma$	$\mu+\beta+\gamma$

α the treatment effect ($T_2 - T_1$),
β the learning (or period) effect,
γ the sequence effect.

Figure 4: Cross-over design with effects.

```
ashinal$id=factor(ashinal$id); ashinal$period=factor(ashinal$period)
ashinalm=lm(pain~treatment+period+id,data=ashinal); anova(ashinalm)
```

Changing the order of factors in the anova formula gives different p-values, because anova performs "sequential tests". To use it correctly, put the factor of interest last in the formula, e.g. treatment, which is "corrected" for the other factors.
Mixed effects model is way better for cross-over design. The R-library lme4 implements the mixed effects models, another library is nlml. The function lmer gives the correct implementation of the crossover design, with the individuals as "random effects".

```
library(lme4); attach(ashinal)
ashinalmer=lmer(pain~treatment+sequence+period+(1|id),REML=FALSE)
summary(ashinalmer)
ashinalmer1=lmer(pain~sequence+period+(1|id),data=ashinal,REML=FALSE)
anova(ashinalmer1,ashinalmer) # test model without treatment inside full model
```

The function lmer does not automatically produce p-values (and they cannot be extracted by anova(ashinalmer)), but these can be found by refitting the model without the effect of interest (in our case treatment), and applying anova with 2 arguments (to test the fit of the reduced model without treatment inside the full model). Factor treatment has a significant effect. Notation: 1 in $(1|id)$ means the the random effect id is with respect to the intercept.

- **Split-plot design:** Select I groups of NJ experimental units randomly from the population. Randomize the I levels of the ("difficult to permute") outer factor over the I groups. Within every group randomize the J levels of the ("easy to permute") inner factor over the NJ units in the group. Perform the experiment NIJ times independently. Example: At two farms (= block) a field was subdivided in 3 parts (= whole plot) and the (outer) factor spray was independently randomized over the 3 whole plots. Next, each of the 3 x 2 = 6 whole plots was subdivided in 2 subplots and within every whole plot the (inner) factor variety was randomized over the 2 subplots.

```
wheat$spray=factor(wheat$spray); wheat$variety=factor(wheat$variety)
wheatlm=lm(yield~spray*variety+farm+farm:spray,data=wheat)
anova(wheatlm)
#output
Df Sum Sq Mean Sq F value Pr(>F)
spray 2 842.17 421.08 76.5606 0.002664 **
variety 1 85.33 85.33 15.5152 0.029157 *
farm 1 456.33 456.33 82.9697 0.002796 **
spray:variety 2 1.17 0.58 0.1061 0.902597
spray:farm 2 15.17 7.58 1.3788 0.376117
Residuals 3 16.50 5.50
```

Interest is in the main and interaction effects of the outer and inner factor. Main effects for spray and variety are significant, whereas interaction effects between these two are not. So, spray causes significant change (0.002) to growth not dependent on crop variety(0.9), and crops also differ between varieties (0.029). Using the spray on different farms does not make a significant difference (0.376). The block factor is the farm with significant difference (0.0027) which is expected.
Using the fixed effects model from above is outdated. rather do mixed effects:

```
wheatlmer=lmer(yield~spray*variety+(1|farm)+(1|farm:spray), data=wheat,REML=FALSE);
wheatlmer1=lmer(yield~spray+(1|farm)+(1|farm:spray),data=wheat,REML=FALSE)
anova(wheatlmer1,wheatlmer) #test for crop variety (left out of simplified model)
#test for all factors:
drop1(wheatlmer, test="F")
```

7 Lecture 7

Contents: 1 contingency tables (1 chisquare test 2 Fisher test), 2 multiple linear regression

- **Contingency table:** an experiment with a count of units in different categories of two factors. We can count for cross-category, columns or rows. For the first one, we have H_0: row variable and column variable are independent. For columns or rows, we have H_0: the distributions over rows / columns factors are equal.

– **Chi-square test:** If we know the total amount of population in the different categories, we can easily calculate the individual entries in the table with the chi-square test, see figure 5. In R, we transform our data in a matrix object and then use the test. Note that each entry in the table should have more than 5 units for a reliable test!

	exact	arts	total
men	?	?	40
women	?	?	20
total	30	30	60

$\Longrightarrow$

	exact	arts	total
men	$60 \cdot \frac{40}{60} \cdot \frac{30}{60}$	$60 \cdot \frac{40}{60} \cdot \frac{30}{60}$	40
women	$60 \cdot \frac{20}{60} \cdot \frac{30}{60}$	$60 \cdot \frac{20}{60} \cdot \frac{30}{60}$	20
total	30	30	60

Figure 5: Chi-square test.

– **Fisher's test:** for a 2x2 table we can calculate the exact p-value without simulation with Fisher's test.

```
#first transform data into a matrix
grades=matrix(c(8,15,13,14,19,15,15,4,7,3,1,4),byrow=TRUE,ncol=3,nrow=4,
dimnames=list(c("A","B","C","D-F"),c("Psychology","Biology","Other")))
z=chisq.test(grades); # if error is given, use simulate.p.value=T
observed(z) #recovers observed values
residuals{z} #which observed values deviate most from H_0?
expected(z) #the expected values
#for a 2x2 matrix we could use Fisher Test:
fisher.test(handed)
```

- **Linear Regression:** an experiment with a numerical outcome Y ("dependent variable") and p numerical explanatory variables X1, . . . , Xp ("independent variables", "predictors"). A linear regression of 1 predictor is the same as a correlation test.
The formula in matrix notation: $Y = X\beta + e$, with β_0 as intercept parameter and other β's as parameters for our predictor variables and e as errors. We want to estimate the β's. We are able to manipulate the predictor variables (take the log of that variable column, or powers, or interactions between two predictors). So, β's are linear but predictors not necessarily. To estimate fit of our regression, we use SSE, sum of squared differences. Coefficient of determination R^2 is calculated as :

$$R^2 = \frac{SS_y - SSE}{SS_y} = \frac{\sum(Y_n - \bar{Y}^2 - \sum \hat{e}_n^2}{\sum(Y_n - \bar{Y}^2}$$

So the estimated variance SSE versus actual variance SS_y. R^2 is the proportion of explained variance. Low R means the model doesn't explain anything (even though p-value could be significant). Call the summary of your model to see p-value and estimated β's. The Estimate columns contains all the β's. The adjusted R^2 adjusts for the number of used predictors.

```
#first investigate the data
plot(sat[,c(1,2,3)]); #scatter matrix plot
# or use:
pairs(sat) #which also gives scatter matrix plot
par(mfrow=c(1,3));
for (i in c(1,2,3)) hist(sat[,i],main=names(sat)[i]) # 3 histograms
datalm = lm(y~x1+...+xp,data=...)
summmary(datalm)
# we can still use the residuals qqnorm and fitted plot for assumptions
qqnorm(residuals(datalm)); plot(fitted(datalm),residuals(datalm))
```

8 Lecture 8

Contents: Strategies to choose predictors (step up & step down), diagnostics in linear regression, problems in linear regression (outliers, influence points, collinearity).

- Step down: begin with full model, delete variable with highest p-value if it is above 0.05, continue.
- Step up: make simple model with only one predictor; check for highest R^2, continue adding predictors to that model if p-value of new predictor is significant.
- Diagnostics for model quality:

  1. Scatter matrix plot Y and each predictor X. With this, you can check linearity for Y vs X but also for some X with another X (possible colinearity).
  2. Scatter plot residuals vs each X in the model, if a pattern is visible, maybe we can transform this X (log, squared etc). Also check the X's not in the model; if pattern is linear, we could include them.
  3. Added variable plot (partial regression plot): plot residuals of Xj against residuals of Y with omitted Xj (to show the effect of adding Xj to the model.) (Or, to show the relationship between Y and Xj , once all other predictors have been accounted for.). Check slope for best fitted line, this indicates the β coefficient for adding that variable. In R:

     ```
     x=residuals(lm(Thigh~Midarm+Triceps))
     y=residuals(lm(Fat~Midarm+Triceps))
     plot(x,y,main="Added variable plot for
     + Thigh", xlab="residual of Thigh",
     + ylab="residual of Fat"))
     ```

  4. Scatter plot of residuals againt Y and $\hat{Y}$, in other terms, residuals vs fitted.
  5. QQplot of residuals (as usual)

  1 is handy for pre-investigation, and in principle we only use **4** and **5** to check model assumptions.

- Outliers are data points with outlying response value (Y). A potential point is an outlying value in the explanatory variable (X). Calculate Cook's distance to see if potential point is influence point:

$$D_i = \frac{1}{(k+1)\hat{\sigma}^2} \sum (Y_{(i),j} - Y_j)^2$$

  With $Y_{(i),j}$ being the predicted response (outcome of model) if we would delete the potential point. If a distance is larger than 1, that point is an influence point.

  ```
  result = cooks.distance(lm(crime$expend~crime$bad + crime$crime +
  crime$lawyers+ crime$employ + crime$pop))
  print(result[result>0.25])
  crime = crime[result<1,] #subset datapoints without the influence points
  round(cor(subset(crime, select = -expend)), 2) #check collinearity
  ```

- For outliers: The mean shift outlier model can be applied to test whether the k-th point significantly deviates from the other points in a linear regression setting.

  ```
  u11=rep(0,16); u11[11]=1; u11
  [1] 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
  forbeslm11=lm(y~x+u11); summary(forbeslm11)
  ```

  If p value for u11 is significant, the outlier is significant.

9 Lecture 9

Content: 1) ANVOCA, 2) prediction and features selection in linear regression with lasso, ridge and elastic net and 3) multiple testing procedures, FDR control.

- ANVOCA is used when one explanatory factor is numerical and not in levels. From this factor it is a-priori evident that it has influence of outcome, and is added to increase the precision of the analysis of the variable in which we are interested in.

```
# a nice plot for ANCOVA related data:
plot(loglongevity~thorax,pch=as.character(activity), data=flies)
# now let's see how ANCOVA works:
fiber1=lm(strength~thickness+type,data=fiber)
anova(fiber1)
# interested in type: thickness is already evident!
# or we use drop1:
drop1(fiber1,test="F") # here all p-values are relevant
# performs strength~thickness+type and
# strength~type+thickness at once
```

- We could plot the best fit line of thickness+type for each level of type (1,2,3). If no interaction exists between thickness and type, then the slope should be the same for each line (the intercept could be different due to differences in type). Or do thickness*type in your lm model.

```
plot(loglongevity~thorax,pch=as.character(activity),data=flies)
abline(lm(loglongevity~thorax,data=flies[flies$activity=="High",]))
abline(lm(loglongevity~thorax,data=flies[flies$activity=="Low",]))
abline(lm(loglongevity~thorax,data=flies[flies$activity=="Isolated",]))
```

- **Lasso and ridge regularization:** add penalty term to high weights. Lasso is L1 regularization; Ridge is L2; elastic net is a combination of both which can be chosen using λ which is ususally selected by cross-validation. Lasso is handy since it puts weights at zero if it thinks the features are not relevant; ridge doesn't do this.

```
library(glmnet)
x=as.matrix(data[,-1]) #remove the response variable
y=as.double(as.matrix(data[,1])) #only the response variable
train=sample(1:nrow(x),0.67*nrow(x)) # train by using 2/3 of the data
x.train=x[train,]; y.train=y[train] # data to train
x.test=x[-train,]; y.test=y[-train] # data to test the prediction quality
lasso.mod=glmnet(x.train,y.train,alpha=1)
cv.lasso=cv.glmnet(x.train,y.train,alpha=1,type.measure='mse')
plot(lasso.mod,label=T,xvar="lambda") #have a look at the lasso path
plot(cv.lasso) # the best lambda by cross-validation
plot(cv.lasso$glmnet.fit,xvar="lambda",label=T)
lambda.min=lasso.cv$lambda.min; lambda.1se=lasso.cv$lambda.1se
coef(lasso.model,s=lasso.cv$lambda.min) #beta's for the best lambda
y.pred=predict(lasso.model,s=lambda.min,newx=x.test) #predict for test
mse.lasso=mean((y.test-y.pred)^2) #mse for the predicted test rows
```

- False Discovery Rate (FDR): the expected proportion of falsely rejected null hypothesis among the rejected hypotheses. V is false positives, R the sum of false positives and true positives, FDR = V/R. In R, we could use p.adjust to adjust p-value taking into account FDR or the older FWER (Family-wise error rate).

```
p.raw=summary(pvcaov)$coef[,4] # vector of individual (raw) p-values
p.raw=p.raw[order(p.raw)] # order the p-values
p.val=as.data.frame(p.raw)
p.val$Bonferroni=p.adjust(p.val$p.raw,method="bonferroni")
```

10 Lecture 10

Contents: 1) logistic regression; 2) Poisson regression.

- **Logistic Regression:** Used for response Y which is categorical. We use sigmoid function to predict. P higher than 0.5, then category 1, otherwise 0.

```
tot=xtabs(~alc+tob,data=esoph) #used to see distribution of datapoints per level
esophglm=glm(cancer~age+age2+alc+tob,data=esoph,family=binomial)
# use family=binomial to create glm (otherwise will be lm)
drop1(model,test="Chisq").
# used to check order of variables; which are significant
predict(glm2,newdata,type="response")
# predict is 1/(1 + e^(-X*Betas))
# some factors might interact so check for that:
glm4=glm(cancer~age*alc,data=esoph,family=binomial)
anova(glm4,test="Chisq") # only the last p-value is relevant
```

- **Poisson regression:** for outcome Y which is a count (number of plants, number of soldiers). Poisson distribution is: $P(Y = k) = \frac{\lambda}{k!}e^{-\lambda}$. High λ (100), the distribution is approximately normal with mean λ and variance λ^2. Poisson is modelled as: $Y = Poisson(\lambda)$, $\lambda = \exp^{lm}$. For each Y, λ is modelled differently thus not from normal distribution thus qqplot and residuals not handy. We use a glm with family=poisson. The model is $\hat{Y} = \hat{\lambda} = \exp^{model}$.

```
galaglm=glm(Species~Area+Elevation+Nearest+Scruz+Adjacent,
+family=poisson,data=gala)
summary(galaglm)
```