Exploratory Data Analysis with R

Visualizing Experimental Designs

Xuemao Zhang
East Stroudsburg University

October 17, 2022

Outline

- Design of Experiments
- Visualizing Completely Randomized Designs
- Visualizing Randomized Complete Block Designss
- Visualizing Factorial Designs
- Visualizing Variance Components
- Visualizing Multiple Comparisons

Design of Experiments

- Two ways of data collection
  - observational study
  - experimental design
- Experiment: people apply some treatment and then observe its effects on the subjects (subjects in an experiments generally are called experimental units)
  - An experiment requires random assignment of subjects to treatments.
  - If done correctly, experiments provide most compelling evidence that a treatment causes an observed outcome
  - For example, in a randomized clinical study patients in the experimental groups receive the drug while patients in the control groups receive a placebo or sugar pill. The patients do not know if they are receiving the experimental treatment or placebo.

Design of Experiments

- Some terminologies:
  - An experimental unit is the object on which a measurement (or measurements) is taken.
  - The response is the variable being measured by the experimenter.
  - A factor is an independent variable whose values are controlled and varied by the experimenter.
  - A level is the intensity setting of a factor.
  - A treatment is a specific combination of factor levels.
- We need several statistical (probabilistic) models in this section.

Visualizing Completely Randomized Design

- Completely randomized design:
    - one factor only; one-way classification.
    - The treatments are assigned completely randomly to the experiment units.
- The data will be like

Trt_1	Trt_2	$\cdots$	Trt_k
y_{11}	y_{21}	$\cdots$	y_{k1}
y_{12}	y_{22}	$\cdots$	y_{k2}
$\vdots$	$\vdots$	$\vdots$	$\vdots$
y_{1,n_1}	y_{2,n_2}	$\cdots$	y_{k,n_k}

Visualizing Completely Randomized Design

- Statistical model: By regarding each n_i observations as a random sample from an infinite population, the probability model is:

$$Y_{ij} = \mu_i + \varepsilon_{ij}, j = 1, 2, \ldots, n_i, i = 1, 2, \ldots, k,$$

  where $\varepsilon_{ij} \sim N(0, \sigma^2)$ are independent normal random errors with common variance σ^2.

- Are the k population means the same, or is at least one mean different from the others?
- We are testing $H_0 : \mu_1 = \mu_2 = \cdots = \mu_k$ versus H_a : At least two means are different from each other or "Not all means are equal". If H_0 is rejected, which treatments are better.

Visualizing Completely Randomized Design

- Example: Is the attention span of children affected by whether or not they had a good breakfast? Twelve children were randomly divided into three groups and assigned to a different meal plan. The response was attention span in minutes during the morning reading time.

No Breakfast	Light Breakfast	Full Breakfast
8	14	10
7	16	12
9	12	16
13	17	15

- The response variable is attention span.
- The experimenter chooses 3 levels of a single factor - breakfast
- Each level of the factor is a treatment
- The experiment is replicated 4 times

Visualizing Completely Randomized Design

- Data

```
resp=c(8,7,9,13,14,16,12,17,10,12,16,15);
Trt=c(rep(1,4),rep(2,4),rep(3,4));
Breakfast=as.data.frame(cbind(resp, Trt));
Breakfast$Trt =factor(Breakfast$Trt); #make "Trt" a factor
str(Breakfast)
```

```
## 'data.frame':    12 obs. of  2 variables:
##  $ resp: num  8 7 9 13 14 16 12 17 10 12 ...
##  $ Trt : Factor w/ 3 levels "1","2","3": 1 1 1 1 2 2 2 2 3 3 ...
```

Visualizing Completely Randomized Design

- Data (or import the data)

```
breakfast=read.csv("../data/breakfast.csv", header = T, sep = ",")
library(tidyr)
pivot_longer(data=breakfast,
        cols=c(No.Breakfast,Light.Breakfast,Full.Breakfast),
  names_to= "treatment", values_to = "resp")->breakfast
str(breakfast)
```

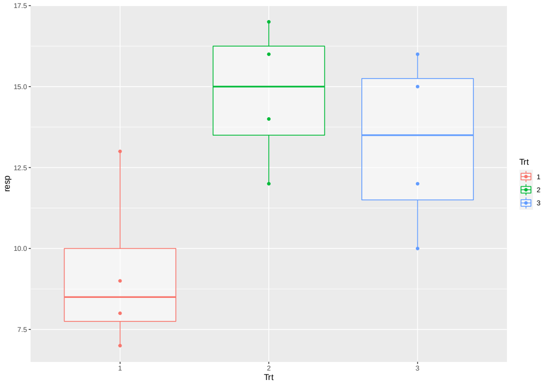
```
## tibble [12 x 2] (S3: tbl_df/tbl/data.frame)
##  $ treatment: chr [1:12] "No.Breakfast" "Light.Breakfast" "Full.B
##  $ resp     : int [1:12] 8 14 10 7 16 12 9 12 16 13 ...
```

```
breakfast$treatment=as.factor(breakfast$treatment)
```

Visualizing Completely Randomized Design

- Box plot and dot plot

```
library(ggplot2)
ggplot(data=Breakfast, aes(x=Trt, y=resp,color=Trt))+
  geom_boxplot(alpha = 0.5)+ #set transparency
  geom_point();
```



Visualizing Randomized Complete Block Designs

- The Blocking Principle
  - Blocking is a technique for dealing with nuisance factors
  - A nuisance factor is a factor that probably has some effect on the response, but it's of no interest to the experimenter... however, the variability it transmits to the response needs to be minimized
- Typical nuisance factors include batches of raw material, operators, pieces of test equipment, time (shifts, days, etc.), different experimental units

Visualizing Randomized Complete Block Designs

Randomized Complete Block Design

- There are two factors: factor of interest and **nuisance factor**.
- A block is a specific level of the **nuisance factor**
- The design uses blocks such that units in each block are relatively similar or homogeneous, with one unit within each block randomly assigned to each treatment (all runs within a block are randomized).
- If the design involves a treatments within each of b blocks, then the total number of observations is ab.
- Variability between blocks can be large, variability within a block should be relatively small
- The purpose of blocking is to isolate the block-to-block variability that might hide/affect the effect of the treatments.
- The design must be balanced: the number of units in each block must be equal to the number of treatments.

Visualizing Randomized Complete Block Designs

- Example: We want to investigate the effect of 3 methods of soil preparation on the growth of seedlings. Each method is applied to seedlings growing at each of 4 locations and the average first year growth is recorded.

Table 1: Seedling Problem

	Location			
Soil Prep	1	2	3	4
A	11	13	16	10
B	15	17	20	12
C	10	15	13	10

- Treatment = soil preparation ($a = 3$)
- Block = location ($b = 4$)
- Is the average growth different for the 3 soil preps?

Visualizing Randomized Complete Block Designs

- Suppose that there are a treatments (factor levels) and b blocks
- Let y_{ij} be the response for the i-th treatment applied to the j-th block. $i = 1, 2, \ldots, a, j = 1, 2, \ldots, b$
- The data will be look like this

Table 2: Data from Block Design

	Treatments			
Blocks	1	2	$\cdots$	a
1	y_{11}	y_{21}	$\cdots$	y_{a1}
2	y_{11}	y_{22}	$\cdots$	y_{a2}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
b	y_{1b}	y_{2b}	13	y_{ab}

Visualizing Randomized Complete Block Designs

- Statistical model:

$$Y_{ij} = \mu + \tau_i + \beta_j + \varepsilon_{ij}, j = 1, 2, \ldots, b, i = 1, 2, \ldots, a,$$

  where $\varepsilon_{ij} \sim N(0, \sigma^2)$ are independent normal random errors with common variance σ^2.

- The relevant null hypothesis for comparing the a treatment means

$$H_0 : \mu_1 = \mu_2 = \ldots = \mu_a$$

Visualizing Randomized Complete Block Designs

- Data

```
growth=c(11,13,16,10,15,17,20,12,10,15,13,10);
location=rep(c(1,2,3,4),3);
soil=c(rep("A",4),rep("B",4),rep("C",4));
seedlings=cbind(growth,location,soil);
seedlings=as.data.frame(seedlings)
seedlings$growth=as.numeric(seedlings$growth)
seedlings$location=as.factor(seedlings$location)
seedlings$soil=as.factor(seedlings$soil)
str(seedlings)
```

```
## 'data.frame':    12 obs. of  3 variables:
##  $ growth  : num  11 13 16 10 15 17 20 12 10 15 ...
##  $ location: Factor w/ 4 levels "1","2","3","4": 1 2 3 4 1 2 3 4
##  $ soil    : Factor w/ 3 levels "A","B","C": 1 1 1 1 2 2 2 2 3 3
```

Visualizing Randomized Complete Block Designs

- Data (or import the data)

```
library(readr)
seedlings = read_csv("../data/Soil.csv")
```

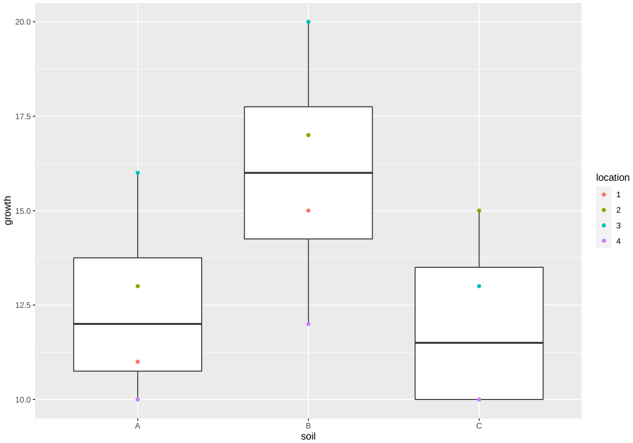
```
## Rows: 12 Columns: 3
## -- Column specification ------------------------------------------
## Delimiter: ","
## chr (1): soil
## dbl (2): growth, location
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
```

```
seedlings$location=as.factor(seedlings$location)
seedlings$soil=as.factor(seedlings$soil)
str(seedlings)
```

```
## spec_tbl_df [12 x 3] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ growth  : num [1:12] 11 15 10 13 17 15 16 20 13 10 ...
##  $ soil    : Factor w/ 3 levels "A","B","C": 1 2 3 1 2 3 1 2 3 1 ...
##  $ location: Factor w/ 4 levels "1","2","3","4": 1 1 1 2 2 2 3 3 3 4 ...
##  - attr(*, "spec")=
```

Visualizing Randomized Complete Block Designss

```
ggplot(data=seedlings, aes(x=soil, y=growth))+
  geom_boxplot()+
  geom_point(aes(color=location));
```



Visualizing Factorial Designs

Two-factor Factorial Design

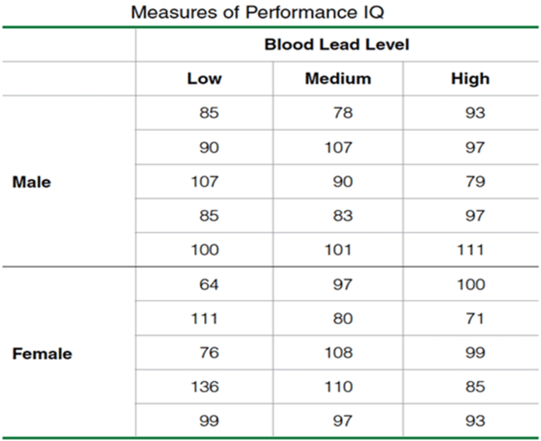
- A two-way classification which involves two factors, both of which are of interest to the experimenter.
- There are *a* levels of factor A and *b* levels of factor B — the experiment is replicated *n* times at each factor-level combination.
- The effect of a factor is defined to be the change in mean response produced by a change in the level of the factor (generally when the factor is changed from low to high)
- The replications in the design allow the experimenter to investigate the interaction between factors A and B.

Visualizing Factorial Designs

Interaction

- There is an interaction between two factors if the effect of one of the factors changes for different levels of the other factor.
- Interaction describes the effect of one factor on the behavior of the other. If there is no interaction, the two factors behave independently.
- Example: The data in the table are categorized with two factors:
  - Sex: Male or Female
  - Blood Lead Level: Low, Medium, or High
- The subcategories are called cells, and the response variable is IQ score. The data are presented on the next slide:

Visualizing Factorial Designs



Measures of Performance IQ

	Blood Lead Level		
	Low	**Medium**	**High**
Male	85	78	93
	90	107	97
	107	90	79
	85	83	97
	100	101	111
Female	64	97	100
	111	80	71
	76	108	99
	136	110	85
	99	97	93

Visualizing Factorial Designs

- Let's explore the IQ data in the table by calculating the mean for each cell and constructing an interaction graph.

```
#Data input
Score=c(85,90,107,85,100,78,107,90,83,101,93,97,79,97,111,
        64,111,76,136,99,97,80,108,110,97,100,71,99,85,93);
Lead1= c(rep(c("low","medium","high"),each=5))
Lead=rep(Lead1,2)
Sex=c(rep(c("M","F"),each=15))
IQ= cbind(Score, Lead, Sex)
IQ=as.data.frame(IQ)
IQ$Score=as.numeric(IQ$Score)
IQ$Lead=as.factor(IQ$Lead)
IQ$Sex=as.factor(IQ$Sex)
```

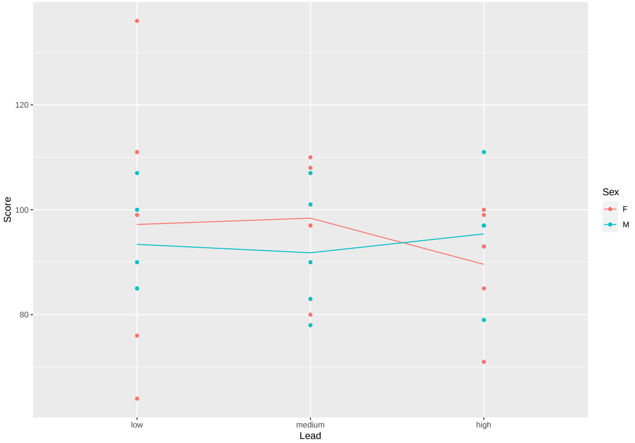
Visualizing Factorial Designs

- Or import the data

```
library(readr)
IQ = read_csv("../data/IQPerformance.csv")
IQ$Lead=as.factor(IQ$Lead)
IQ$Sex=as.factor(IQ$Sex)
str(IQ)
```

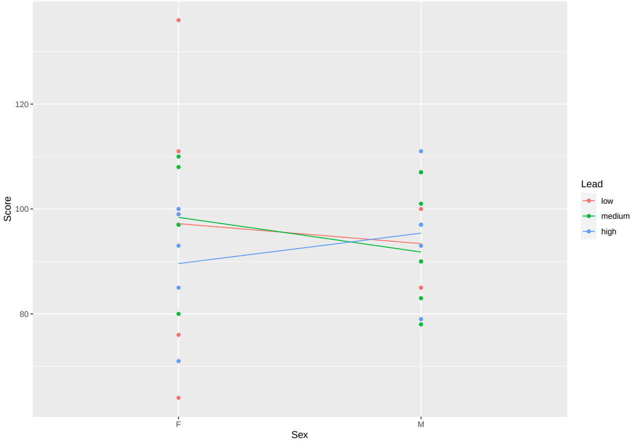
Visualizing Factorial Designs

```
IQ$Lead=factor(IQ$Lead, levels=c("low","medium","high"))
library(ggplot2);
ggplot(IQ, aes(x=Lead, y=Score, group=Sex, col=Sex)) +
  geom_point() + stat_summary(fun= mean, geom = "line");
```



Visualizing Factorial Designs

```
ggplot(IQ, aes(x=Sex, y=Score, group=Lead, col=Lead)) +
  geom_point() + stat_summary(fun= mean, geom = "line");
```



Visualizing Factorial Designs

- Numerical summary

```
library(dplyr);
IQ%>%group_by(Lead, Sex)%>%summarise(mvalue=mean(Score))->IQ_mean;
IQ_mean
```

```
## # A tibble: 6 x 3
## # Groups:   Lead [3]
##   Lead   Sex   mvalue
##   <fct>  <fct>  <dbl>
## 1 low    F       97.2
## 2 low    M       93.4
## 3 medium F       98.4
## 4 medium M       91.8
## 5 high   F       89.6
## 6 high   M       95.4
```

2-Way ANOVA for Factorial Design

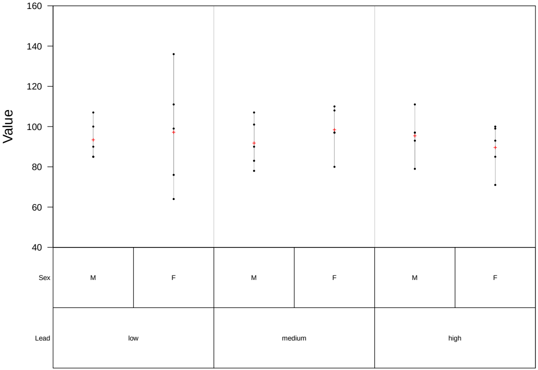
Interaction

- An interaction effect is suggested if the line segments are far from being parallel.
- No interaction effect is suggested if the line segments are approximately parallel.
- For the IQ scores, it appears there is an interaction effect:
    - Females with high lead exposure appear to have lower IQ scores, while males with high lead exposure appear to have high IQ scores.

Visualizing Variance Components

- When there are two or more factors, we may want to check the spread of data due to the factors - Variance Components
- The R-package VCA is used to perform variance component analysis.

```
library(VCA);
IQ=as.data.frame(IQ) #the data must be a dataframe
varPlot(form=Score~Lead+Sex, Data=IQ)
```



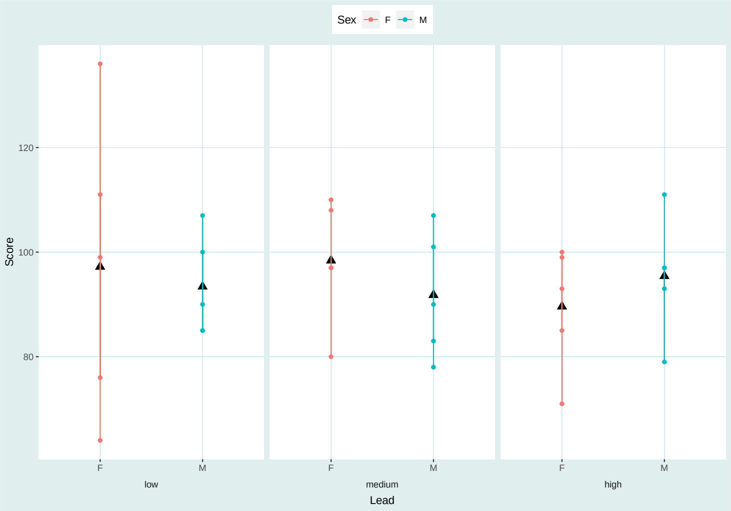
Visualizing Variance Components

- Let's try using ggplot

```
IQ%>%group_by(Lead,Sex) %>% mutate(y_mean=mean(Score)) %>%
ggplot(aes(x=Sex,y=Score,color=Sex))+
  geom_point()+
  geom_point(aes(x=Sex,y=y_mean), pch=17,color="black",size=3)+
  geom_line()+
  facet_grid(~Lead,scales="free_x",space="free_x",switch="x")+
  labs(x="Lead")+
  theme(legend.position = "top",
    plot.background = element_rect(fill="azure2"),
    panel.background = element_rect(fill = "white"),
    panel.grid.major = element_line(size = 0.2,
              linetype = 'solid',colour = "lightblue"))+
  theme(strip.placement = "outside",
      # Place facet labels outside x axis labels
      strip.background = element_rect(fill = "azure2"))
```

Visualizing Variance Components

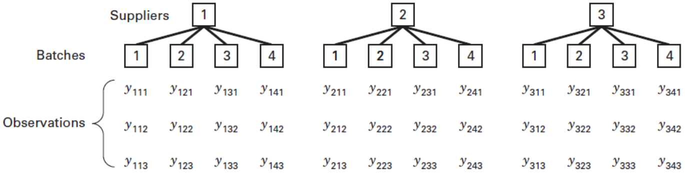
- Let's try using `ggplot`



Visualizing Variance Components

Nested Designs

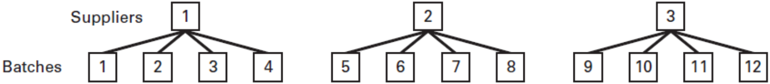
- In certain multifactor experiments, the levels of one factor (e.g., factor B) are similar but not identical for different levels of another factor (e.g., A). Such an arrangement is called a **nested, or hierarchical**, design, with the levels of factor B nested under the levels of factor A.
- For example, consider a company that purchases its raw material from three different suppliers. The company wishes to determine whether the purity of the raw material is the same from each supplier.
  - There are four batches of raw material available from each supplier, and three determinations of purity are to be taken from each batch.


Suppliers
1
2
3
Batches
1 2 3 4
1 2 3 4
1 2 3 4
Observations
y_{111} y_{121} y_{131} y_{141} y_{211} y_{221} y_{231} y_{241} y_{311} y_{321} y_{331} y_{341}
y_{112} y_{122} y_{132} y_{142} y_{212} y_{222} y_{232} y_{242} y_{312} y_{322} y_{332} y_{342}
y_{113} y_{123} y_{133} y_{143} y_{213} y_{223} y_{233} y_{243} y_{313} y_{323} y_{333} y_{343}

Visualizing Variance Components

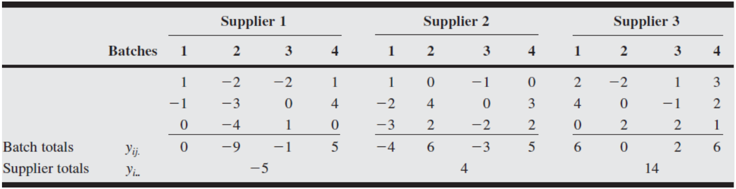
Nested Designs

- This is a **two-stage nested design**, with batches nested under suppliers.
  - The number of stages can be more than two. See `VCAdata1` in the `VCA` package of a three-stage nested design
- Batch 1 from supplier 1 has no connection with batch 1 from any other supplier, batch 2 from supplier 1 has no connection with batch 2 from any other supplier, and so forth.
- To emphasize the fact that the batches from each supplier are different batches, we may renumber the batches as 1, 2, 3, and 4 from supplier 1; 5, 6, 7, and 8 from supplier 2; and 9, 10, 11, and 12 from supplier 3,



Visualizing Variance Components

Nested Designs



		Supplier 1				Supplier 2				Supplier 3			
	Batches	1	2	3	4	1	2	3	4	1	2	3	4
		1	−2	−2	1	1	0	−1	0	2	−2	1	3
		−1	−3	0	4	−2	4	0	3	4	0	−1	2
		0	−4	1	0	−3	2	−2	2	0	2	2	1
Batch totals	$y_{ij.}$	0	−9	−1	5	−4	6	−3	5	6	0	2	6
Supplier totals	$y_{i..}$	−5				4				14			

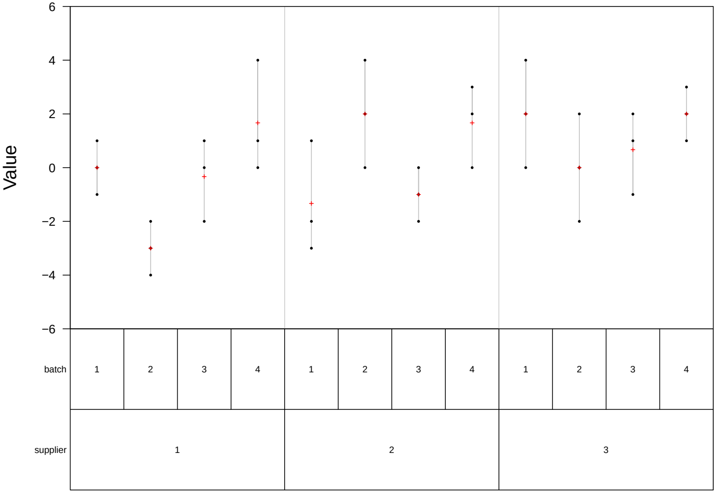
```
##  supplier 1  supplier 2   supplier 3
y=c(1, -1, 0,     1, -2, -3,   2, 4, 0, ## batch 1
    -2, -3, -4,  0, 4, 2,      -2, 0, 2,## batch 2
    -2, 0, 1,     -1, 0, -2,   1, -1, 2,## batch 3
    1, 4, 0,      0, 3, 2,     3, 2, 1);## batch 4
purity=data.frame(y=y, batch=factor(rep(1:4, each=9)),
                supplier=factor(rep(rep(1:3, each=3), 4)));
str(purity);
```

```
## 'data.frame':    36 obs. of  3 variables:
##  $ y       : num  1 -1 0 1 -2 -3 2 4 0 -2 ...
##  $ batch   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 2 ...
##  $ supplier: Factor w/ 3 levels "1","2","3": 1 1 1 2 2 2 3 3 3 1 ...
```

Visualizing Variance Components

Nested Designs

```
varPlot(form=y~supplier/batch, Data=purity);#batch is nested in supp
```



Visualizing Multiple Comparisons

There are several informal methods for determining which means are different:

- Construct boxplots of the different samples to see if one or more of them is very different from the others.
- Construct confidence interval estimates of the means for the different samples, then compare those confidence intervals to see if one or more of them does not overlap with the others (pairwise comparison) or conduct pairwise hypotheses. The method is called LSD (Least Significant Difference).
    - LSD problem: In hypotheses test problems involving a single null hypothesis H_0 the statistical tests are often chosen to control the Type I error rate of incorrectly rejecting H_0 at a pre-specified significance level α. In general, when testing m null hypotheses using independent test statistics, the probability of committing at least one Type I error is $1-(1-\alpha)^m$.

Visualizing Multiple Comparisons

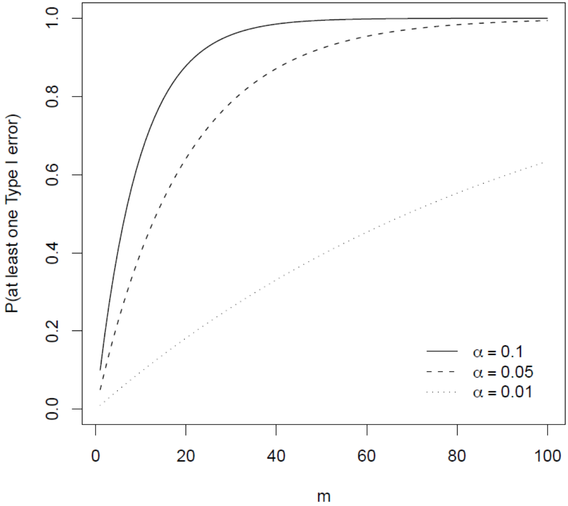


Figure 1: Probability of committing at least one Type I error for diferent significance levels α and number of hypotheses *m*.

Visualizing Multiple Comparisons

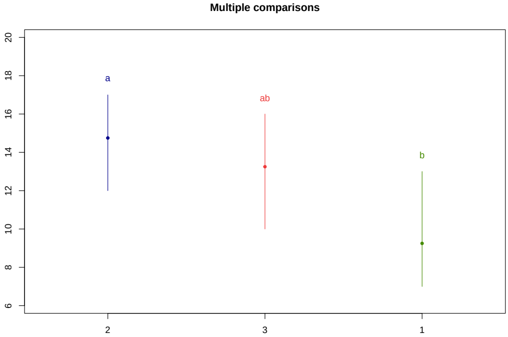
Several Methods for multiple comparisons:

- Bonferroni test: run t-test on all pairs of hypothesis tests, calculate the p-values and apply a p-value correction for multiple testing problems.
  - Holm procedure: a step-down procedure, which basically consists of repeatedly applying Bonferroni's method while testing the hypotheses in a data-dependent order.
- Tukey's test: also referred to as the Tukey HSD (Honest Significance Difference) test, controls for the Type I error rate across multiple comparisons.
  - Tukey's test is based on the studentized range. In essence, the Tukey test takes the maximum over the absolute values of all pairwise test statistics.
- Dunnett test(Many-to-one): the standard method for the the classical many-to-one problem of comparing several groups with a common **control group**.

Visualizing Multiple Comparisons

- To visualizing multiple comparisons at significance level α, we must first fit a statistical linear model.
- We use package agricolae
  - For the breakfast problem

```
library(agricolae);
Model1=lm(resp ~ Trt, data= Breakfast);
compmeans=HSD.test(aov(Model1),"Trt",alpha=0.05,group=TRUE);
plot(compmeans, main="Multiple comparisons"); box();
```



Visualizing Multiple Comparisons

Dunnett test

- Example: A company developed specialized heating blankets designed to help the body heat following a surgical procedure. Four types of blankets b_0, b_1, b_2 and b_3 were tested on surgical patients to assess recovery times. The blanket b_0 was a standard blanket already in use at various hospitals. The primary outcome of interest was recovery time in minutes of patients allocated randomly to one of the four treatments. Lower recovery times would indicate a better treatment effect.

- The `glht` function from `multcomp` takes the fitted response model to perform the multiple comparisons

- We can use the `confint` method associated with the `glht` function to visualize the confidence intervals.

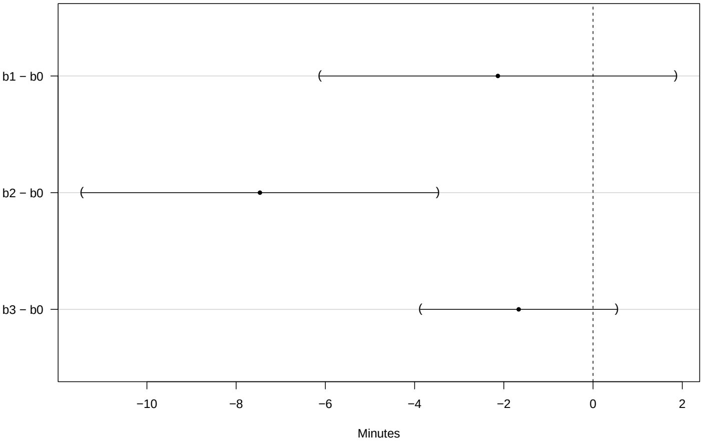
Visualizing Multiple Comparisons

```
library(multcomp)
data(recovery)
siglevel= 0.05
recovery$blanket <- relevel(recovery$blanket, ref = "b0");
# b0 is set as reference level
recovery.aov <- aov(minutes ~ blanket, data = recovery)
recovery.mc <- glht(recovery.aov,linfct = mcp(blanket="Dunnett"),
                    alternative = "two.sided")
#the mcp function for the linfct argument is used
#to specify the comparisons type
summary(recovery.mc);
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Dunnett Contrasts
##
##
## Fit: aov(formula = minutes ~ blanket, data = recovery)
##
## Linear Hypotheses:
##              Estimate Std. Error t value Pr(>|t|)
```

Visualizing Multiple Comparisons

```
recovery.ci <- confint(recovery.mc, level = 1-siglevel)
plot(recovery.ci, main = "", ylim = c(0.5, 3.5), xlab = "Minutes")
```



License