Summary for Demos

周睿

2022-12-14

目录

1 清除当前工作内存

```
rm(list = ls())
```

2 查看和设定路径

```
getwd()
```

```
## [1] "/Users/zhourui/Desktop/心理统计/demos"
```

```
# setwd('desigated path')
```

3 读取指定文件

- csv 文件：r 原生函数 `read.csv`
- excel 文件：调用包 `readxl`
- sav 文件：调用包 `haven`（通常读取的是来自 SPSS 文件）

```
# csv
d13=read.csv('demo 13.csv')
```

```
## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## line 1 appears to contain embedded nulls
```

```
## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## line 2 appears to contain embedded nulls
```

```
## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## line 3 appears to contain embedded nulls
```

```
## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## line 4 appears to contain embedded nulls
```

```
## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## line 5 appears to contain embedded nulls
```

```
## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## incomplete final line found by readTableHeader on 'demo 13.csv'
```

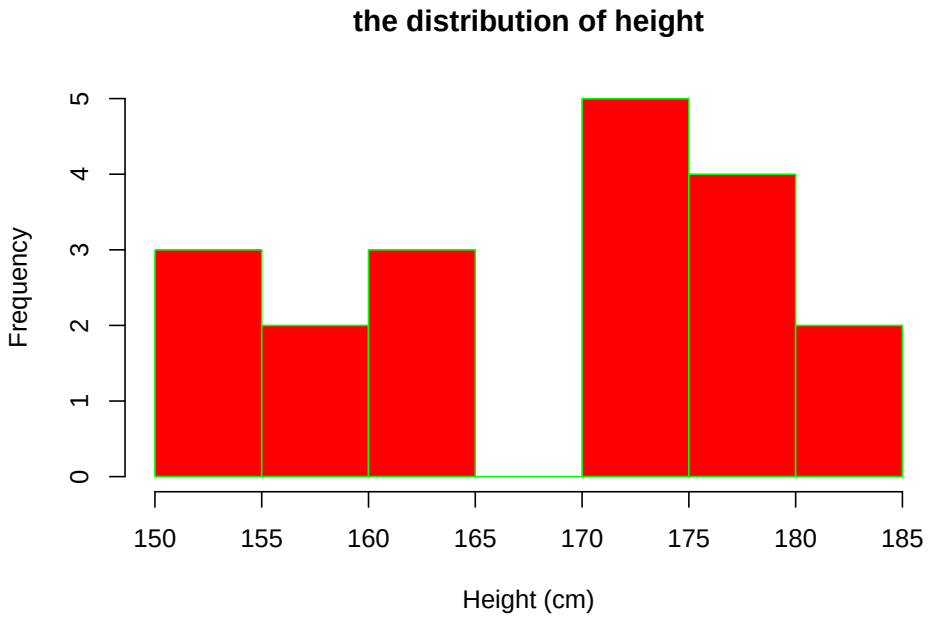
```
# excel
library(readxl)
d1=read_excel('demo 1.xlsx',sheet = 1)

# sav
library(haven)
d4=read_sav('demo 4.sav')
```

4 基本绘图

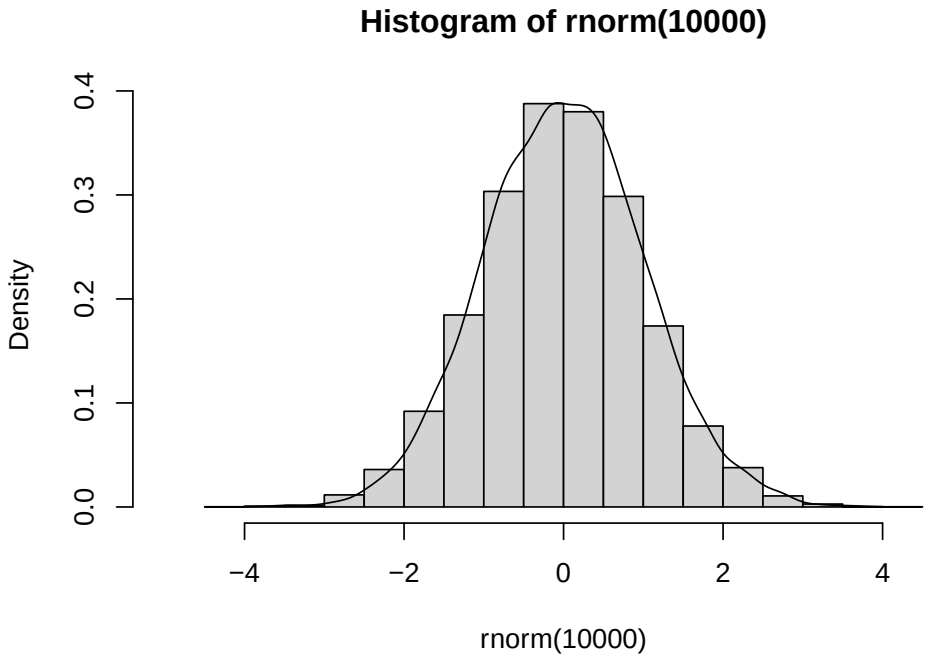
- 直方图：hist
- 多边形图：polygon
- 棒图（条形图）：barplot
- 茎叶图：stem

- 通用：plot

```
age <- d1$age
height <- d1$height
weight <- d1$weight

# histogram
hist(height, main="the distribution of height", xlab= "Height (cm)", col="red", border=
```



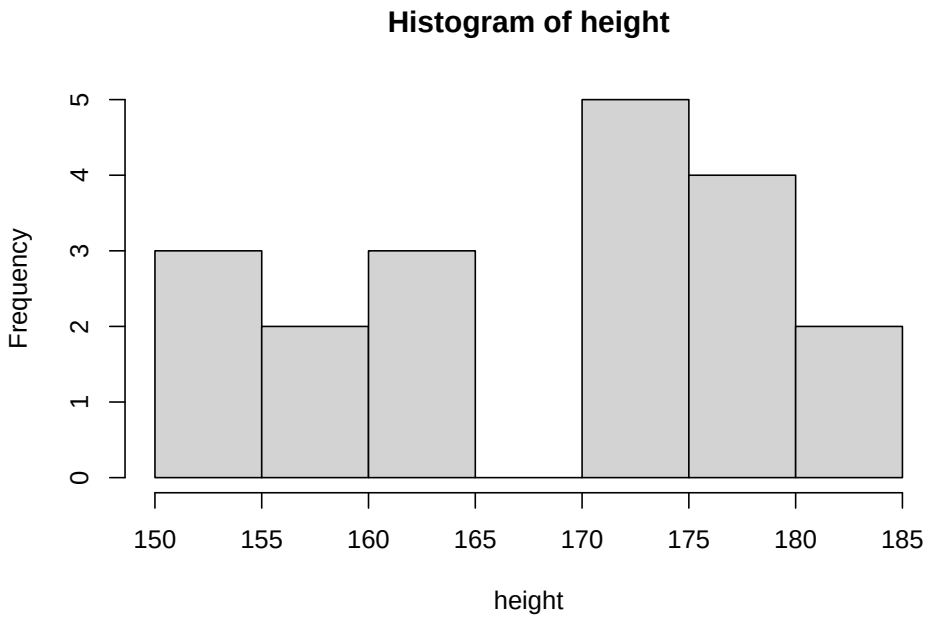

```
set.seed(1213)
hist(rnorm(1e4),xlim = c(-5,5),freq = F)
lines(density(rnorm(1e4)))
```



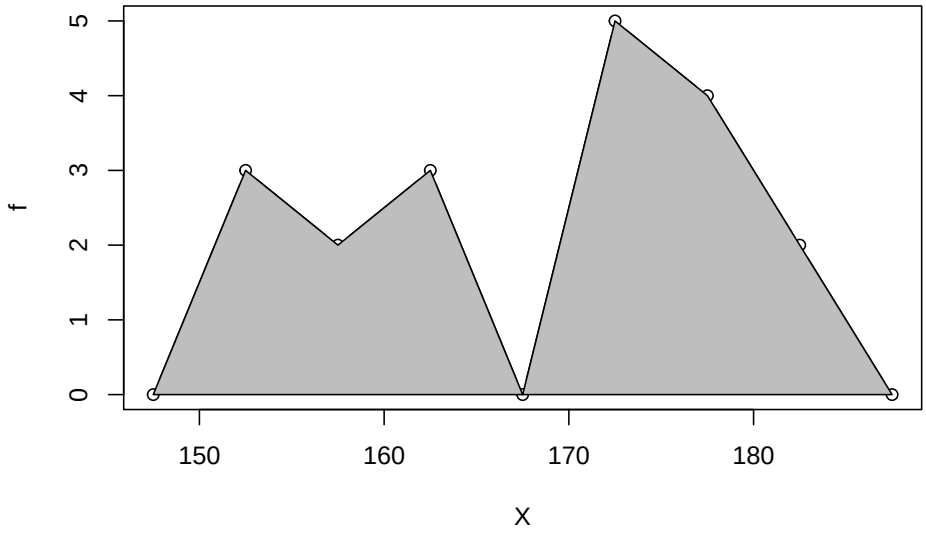

```
# 茎叶图
stem(height)
```

```
##
##   The decimal point is 1 digit(s) to the right of the |
##
##   15 | 14567
##   16 | 344
##   17 | 3444589
##   18 | 0035
```

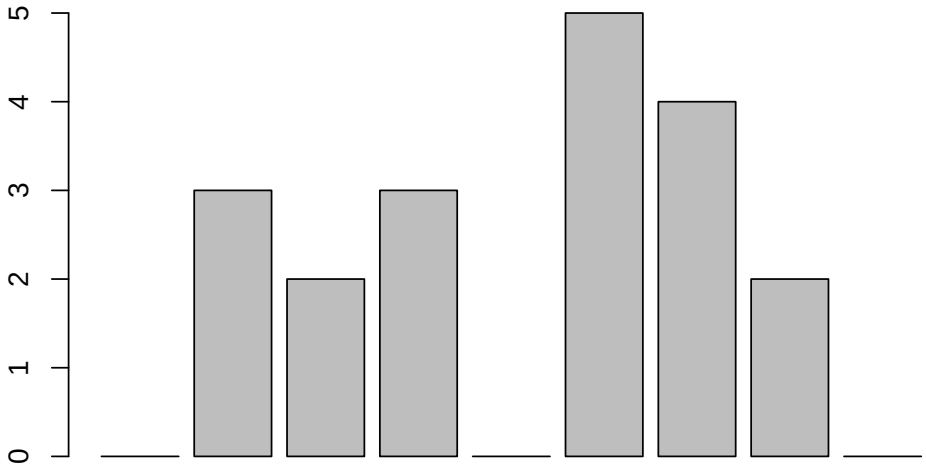
```
# polygon
hist_object = hist(height) # 这样能够调用 hist 结果的某些属性，比如 breaks，counts 等
```



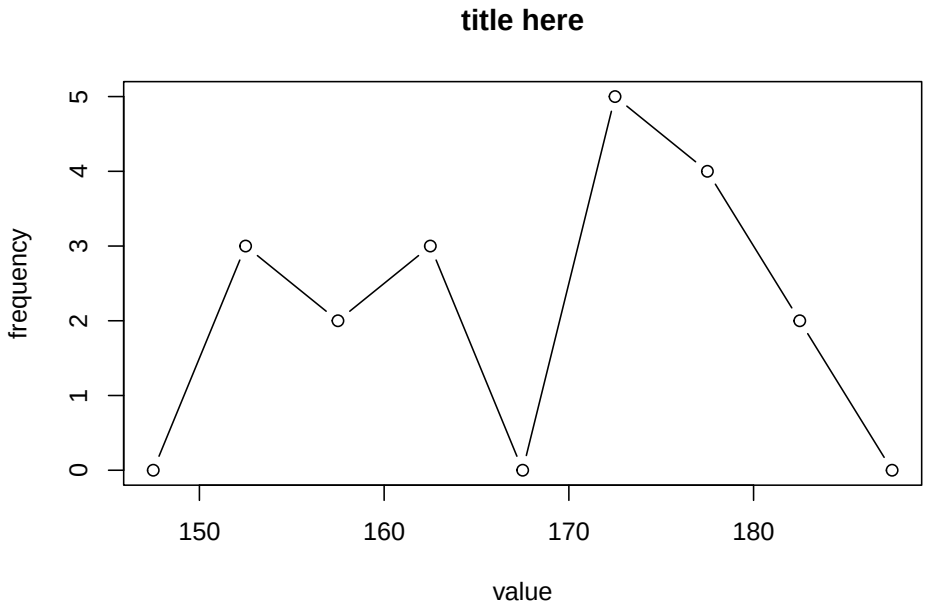

```
f = hist_object$counts
X = hist_object$mids
## 根据定义，多边形图的左右两侧分别要多画一个封闭的点
stepSize = X[2]-X[1]
X = c(X[1]-stepSize, X, X[length(X)]+stepSize)
f = c(0, f, 0)
## polygon 函数相当于是将各个点闭环连接，但首先要有 plot 对象
plot(X,f,"b") # 如果单纯想让结果不额外受影响，可以指定 type='n'
polygon(X,f,col = 'grey') # 颜色默认为空，那么便只是连接各点
```




```
# 棒图
barplot(f)
```




```
# 简单图
plot(X,f,"b",main = "title here",xlab="value",ylab="frequency")
```



5 基本处理与运算

- +, -, *, /, ^, %%(求余), %/%(求商的整数部分)
- sum, prod：向量元素和、乘积
- sort：排序
- abs, sqrt, factorial: 绝对值，平方根，阶乘
- log, exp, log10, log2: 对数与指数函数
- <，>，<=，>=，==，!=：比较运算符
- range, max, min: 全距，最大值、最小值
- floor, ceiling: 向下取整与向上取整
- var, sd, mean, median, quantile: 方差、标准差、均值、中位数、分位数
- cat, print,sprintf,paste: 显示对象

```
mean(age) # 计算 age 这个变量的平均值
```

```
## [1] 18.10526
```

```
median(age)# 计算 age 这个变量的中位数
```

```
## [1] 20
```

```
sd(age)# 计算 age 这个变量的标准差 (sample)
```

```
## [1] 4.214983
```

```
var(age)# 计算 age 这个变量的方差 (sample)
```

```
## [1] 17.76608
```

```
range(age)# 返回 age 这个变量的范围
```

```
## [1] 11 23
```

```
quantile(age)  # 百分位数
```

```
##   0%  25%  50%  75% 100%
##   11   15   20   21   23
```

```
# 计算众数
## 特殊的是，r 语言本身没有直接计算众数的函数
table(age) # 能够得到名称（列索引）为每个值，第一行为每个值出现次数的 table
```

```
## age
## 11 12 14 16 18 19 20 21 22 23
##  3  1  1  1  1  1  4  3  3  1
```

```
max(table(age)) # 获取的是出现次数最大的
```

```
## [1] 4
```

```
which.max(table(age)) # 返回值最大的元素
```

```
## 20
##  7
```

```
as.numeric(names(which.max(table(age)))) # 获取其索引（names），并将其转变为数字类型
```

```
## [1] 20
```

6 数据筛选

- Series[condition]
- DataFrame[condition for Series,]：相当于对列全选
- ifelse(condition,yes,no)

```
mean(age)
```

```
## [1] 18.10526
```

```
mean(age[weight>50])
```

```
## [1] 19.2
```

```
mean(weight[age==20])
```

```
## [1] 41
```

```
d1[d1$age>median(d1$age),]
```

```
## # A tibble: 7 x 3
##     age weight height
##   <dbl>  <dbl>  <dbl>
## 1    21     41    155
## 2    22     44    157
## 3    21     55    174
## 4    22     56    151
## 5    22     51    180
## 6    21     50    173
## 7    23     32    179
```

```
d1[d1$age>median(d1$age),c(2,3)]
```

```
## # A tibble: 7 x 2
##   weight height
##    <dbl>  <dbl>
## 1     41    155
## 2     44    157
## 3     55    174
## 4     56    151
## 5     51    180
## 6     50    173
## 7     32    179
```

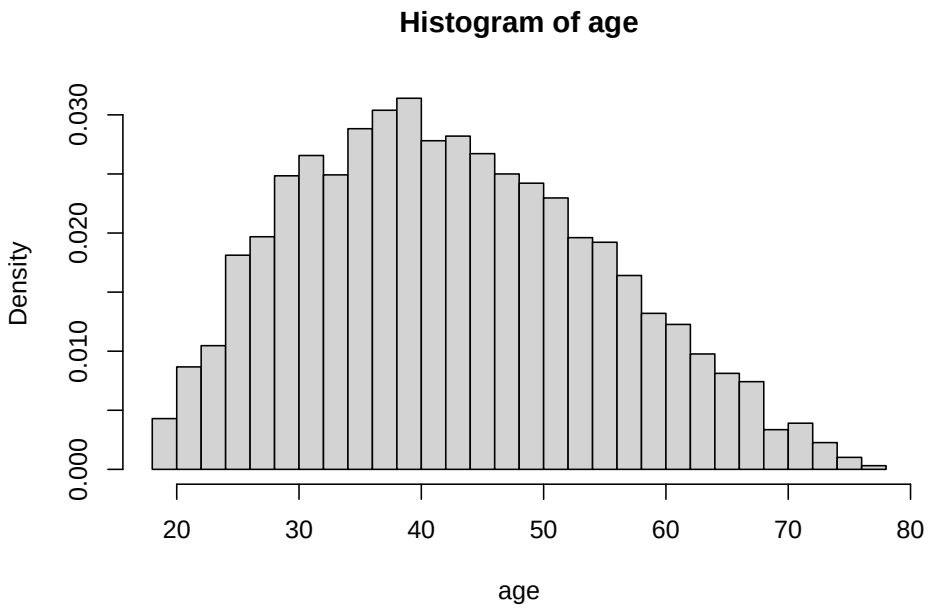
7 因子类型

```
# 定义名义和顺序变量，用 factor()
status <- c("poor","improved","excellent","poor")
status <- factor(status)
## Levels: excellent improved poor
status <- factor(status, levels = c("poor","improved","excellent"))
## Levels: poor improved excellent
status <- factor(status, ordered = T,levels = c("poor","improved","excellent"))
## Levels: poor < improved < excellent
```

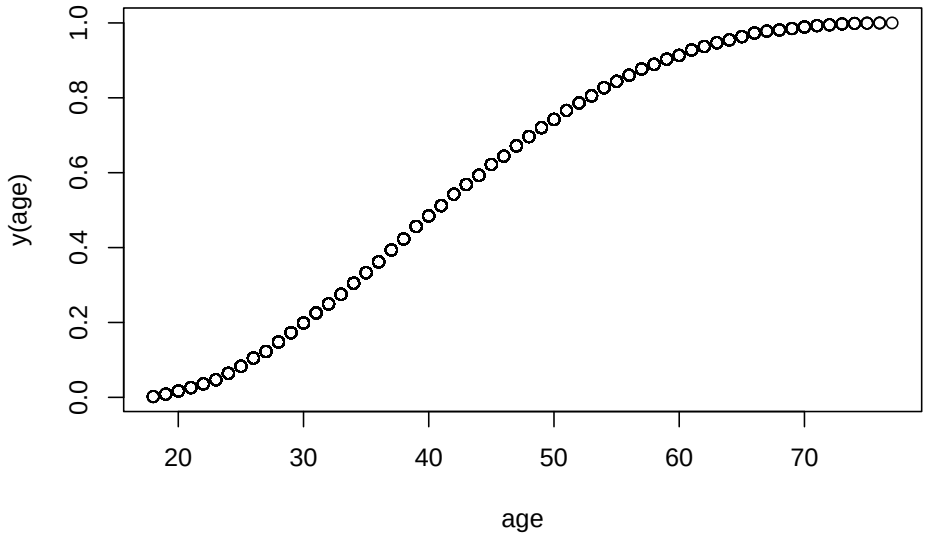
8 频率分布图/累计分布图

```
age=d4$age

# 概率密度图（即频率分布直方图）
hist(age,
```

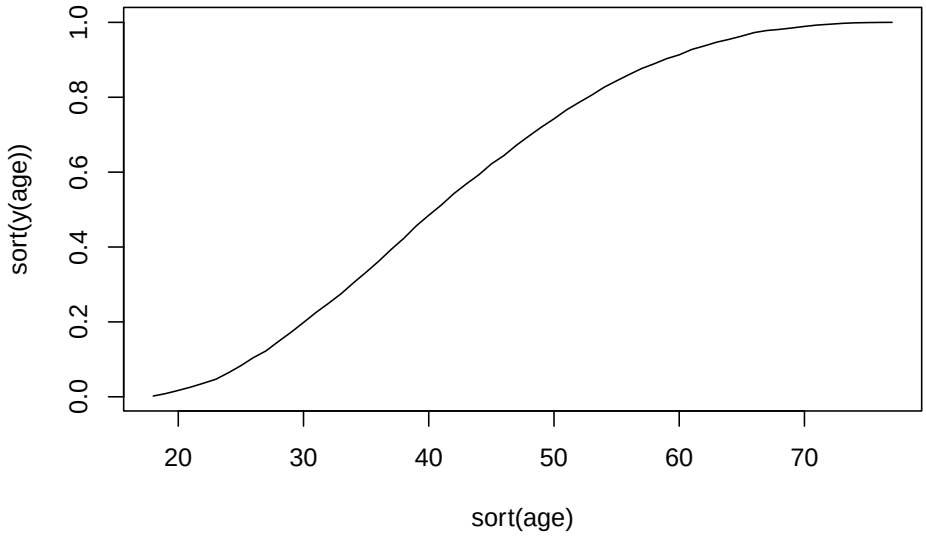
```
    freq = FALSE,
    breaks = seq(18, 78, by = 2),
    right = FALSE)
```




```
# 累积分布图
y <- ecdf(age)
plot(age,y(age)) # 散点图
```




```
plot(sort(age),sort(y(age)),
     type = "l") # 累积分布曲线（实际上是折线图）
```



9 对分布的检验

在直方图呈现的基础上，有方向性地考察分布情况

通常加载 moments 包，内含偏度、峰度、偏态检验等函数

```
library(moments)
skewness(age) # 偏度
```

```
## [1] 0.299057
```

```
kurtosis(age) # 峰度
```

```
## [1] 2.397496
```

```
agostino.test(age) # 检验是否正态（对称）分布
```

```
##
##  D'Agostino skewness test
##
## data:  age
## skew = 0.29906, z = 9.57285, p-value < 2.2e-16
## alternative hypothesis: data have a skewness
```

```
agostino.test(age,alternative = "less")  # 是否 negative skew 分布
```

```
##
##  D'Agostino skewness test
##
## data:  age
## skew = 0.29906, z = 9.57285, p-value < 2.2e-16
## alternative hypothesis: data have positive skewness
```

```
agostino.test(age,alternative = "greater")  # positive skew?
```

```
##
##  D'Agostino skewness test
##
## data:  age
## skew = 0.29906, z = 9.57285, p-value = 1
## alternative hypothesis: data have negative skewness
```

```
# 正态性检验是 r 语言自带的
shapiro.test(age[1:5000])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  age[1:5000]
## W = 0.98248, p-value < 2.2e-16
```

10 概率密度或者累计概率计算

- 分布类型：norm,binom,unif,exp,t,f,chisq
- 计算函数：p,q,r,d

```
# 计算年龄的 z 分数
age_zscore <- scale(age)
## 同时学会 z 分数的转换，给定均值和标准差

# 正态分布
dnorm(1.96) # 已知 z 分数计算【概率密度】，缺省就是直接用 Z 分数来算，也可以指定 mean and s
```

```
## [1] 0.05844094
```

```
pnorm(696, mean = 500, sd = 100) # 计算正态曲线下，某 X 值左侧的累积概率
```

```
## [1] 0.9750021
```

```
pnorm(1.96, lower.tail = FALSE) # 计算标准正态曲线下，某 z 分数右侧的累积概率
```

```
## [1] 0.0249979
```

```
qnorm(0.975) # 已知累积概率，求 z 分数
```

```
## [1] 1.959964
```

```
rnorm(50, mean = 20, sd = 3) # 从满足某正态分布的数据中随机抽取 50 个数
```

```
##  [1] 19.35725 24.58279 17.01680 20.77258 18.49307 21.02807 18.22334 19.66128
##  [9] 23.09224 20.96562 16.20613 20.35738 21.45413 17.71007 20.96260 17.61625
## [17] 21.62569 20.23359 25.58374 22.96528 23.47854 15.70209 21.49335 18.37056
## [25] 22.16432 18.61622 14.89903 25.17583 19.75481 20.69772 20.76265 21.75942
## [33] 24.17083 22.51428 15.63724 16.43384 24.64935 24.33550 26.07293 24.21745
## [41] 14.57082 19.53327 14.39005 24.86466 22.67790 21.72658 21.22387 20.13683
## [49] 20.49089 22.68040
```

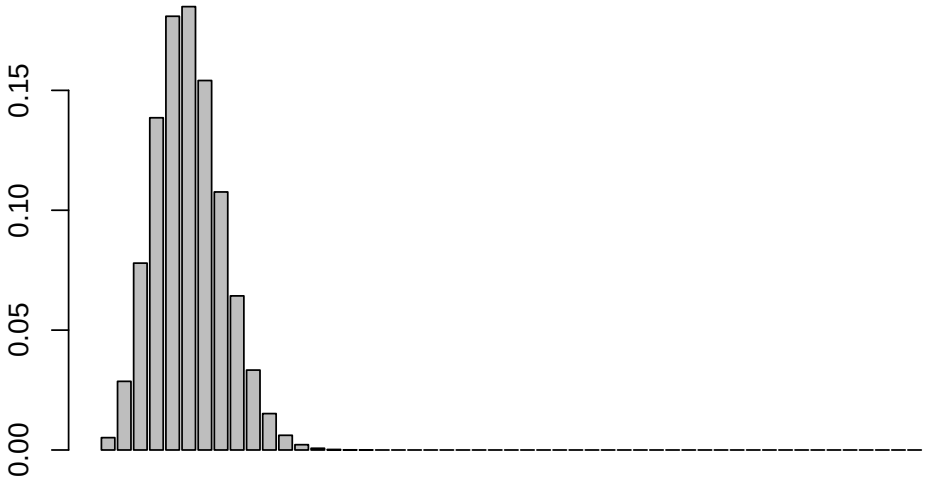
```
# 二项分布（假设抽 50 次奖，每次中奖概率均为 0.1）
dbinom(3, size = 50, prob = 0.1) # 50 次中抽中 3 次的概率
```

```
## [1] 0.1385651
```

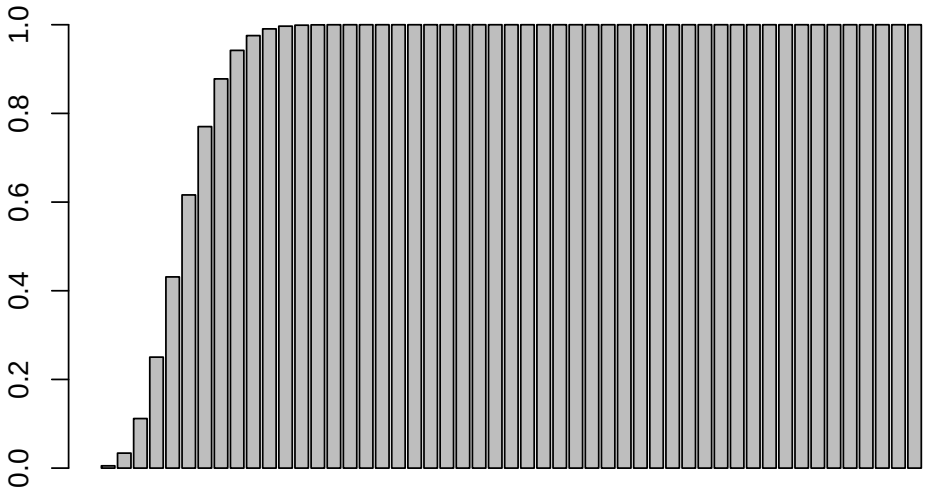
```
pbinom(3, size = 50, prob = 0.1) # 50 次中至少抽中 3 次的概率
```

```
## [1] 0.2502939
```

```
barplot(dbinom(c(0:50), size = 50, prob = 0.1)) # 抽中 0 次-抽中 50 次的概率分布图
```

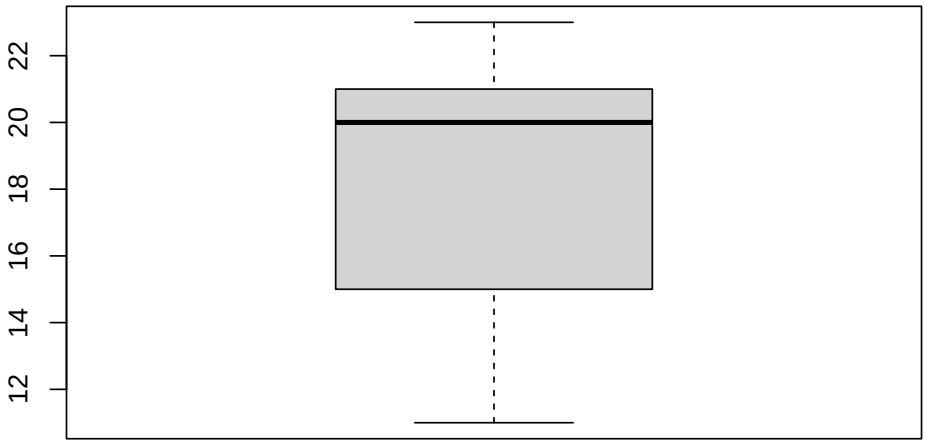


```
barplot(pbinom(c(0:50), size = 50, prob = 0.1)) # 抽中 0 次-抽中 50 次的累积概率分布图
```

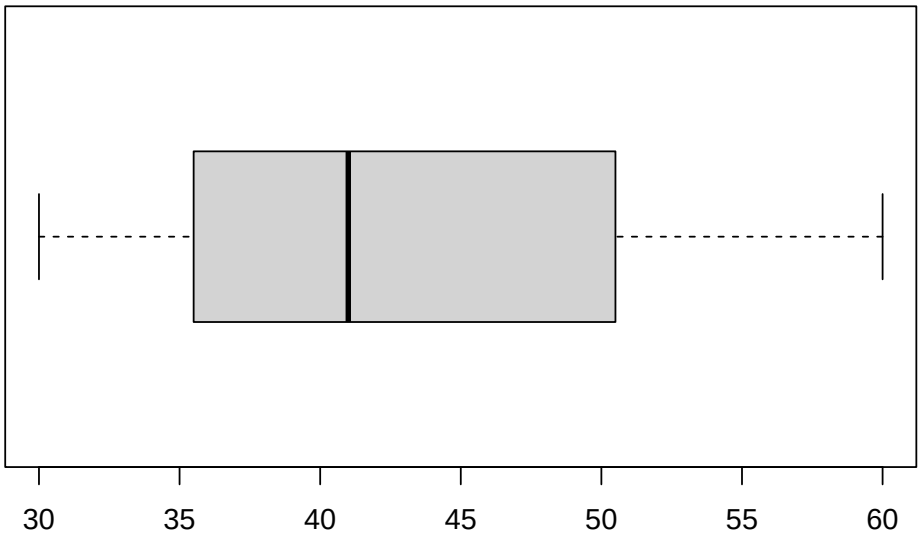


箱型图

```
age=d1$age
boxplot(age)
```

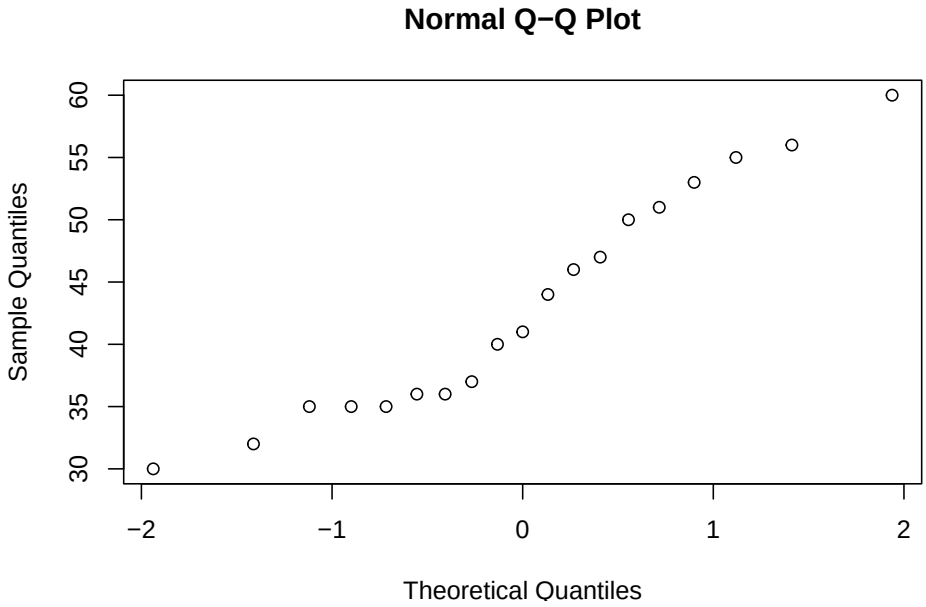



```
boxplot(weight,horizontal = TRUE)
```

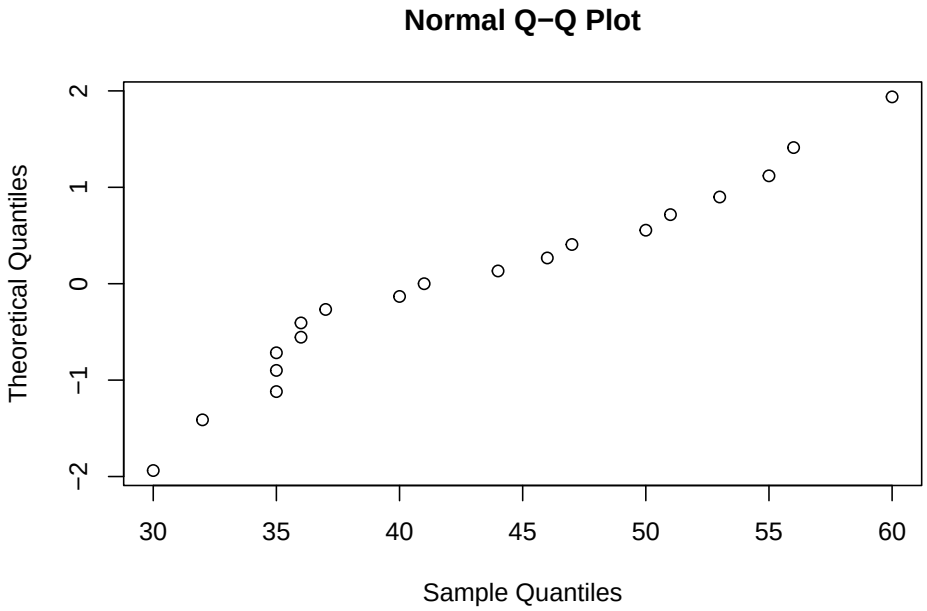


11 QQ 图

```
# QQ plot for testing against normal distribution
qqnorm(weight)
```

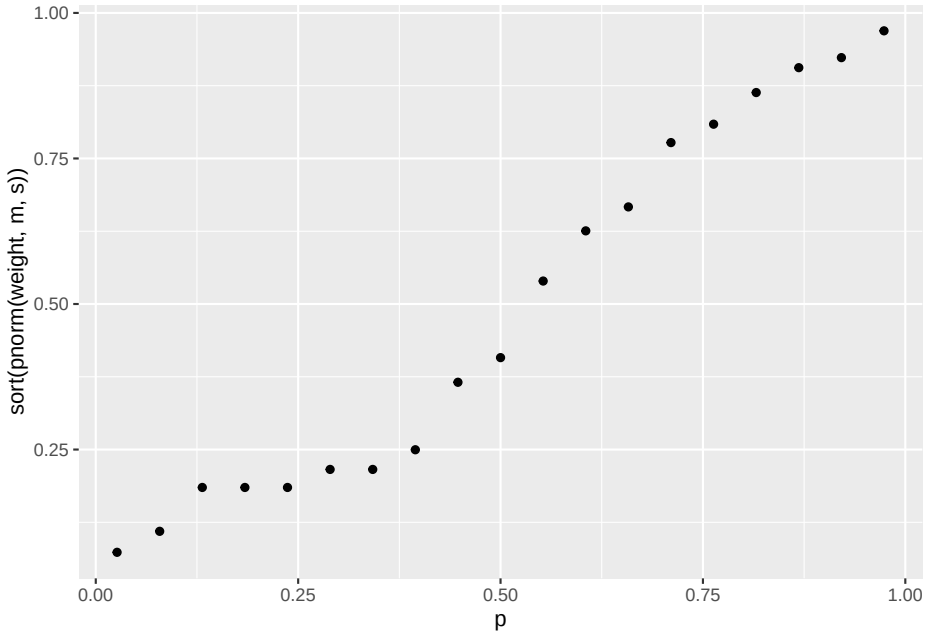


```
qqnorm(weight,datax=TRUE) # 把数据放在横轴上
```



12 PP 图

```
#PP plot
library(ggplot2)
m <- mean(d1$weight)
s <- sd(d1$weight)
n <- nrow(d1)
p <- (1 : n) / n - 0.5 / n
ggplot(d1) + geom_point(aes(x = p, y = sort(pnorm(weight, m, s))))
```



13 t 检验

- 单样本 t 检验
- 双样本 t 检验
  - 配对：`paired=T`
  - 独立：`paired=F`(default)
    - 方差同质：`var.equal=T`
    - 方差不同质：`var.equal=F`
      - 方差同质性检验：`car::leveneTest()`

```
d71=read_excel('demo 7-1.xlsx')
age=d71$age
x1=d71$weight
x2=d71$`weight-after`
d72=read_excel('demo 7-2.xlsx')
x=d72$weight
```

```r
# one-sample t-test
t.test(age,mu=20) # 双尾检验
```

```
##
##  One Sample t-test
##
## data:  age
## t = -1.9594, df = 18, p-value = 0.06574
## alternative hypothesis: true mean is not equal to 20
## 95 percent confidence interval:
##  16.07371 20.13682
## sample estimates:
## mean of x
##  18.10526
```

```r
t.test(age,mu=22,alternative = "less") # 单尾检验 <22
```

```
##
##  One Sample t-test
##
## data:  age
## t = -4.0277, df = 18, p-value = 0.0003948
## alternative hypothesis: true mean is less than 22
## 95 percent confidence interval:
##      -Inf 19.78207
## sample estimates:
## mean of x
##  18.10526
```

```r
t.test(age,mu=15,alternative = "greater",conf.level = .99) # 单尾检验 >22, 置信区间为 99
```

```
##
##  One Sample t-test
##
## data:  age
```

```
## t = 3.2113, df = 18, p-value = 0.00242
## alternative hypothesis: true mean is greater than 15
## 99 percent confidence interval:
##  15.63715      Inf
## sample estimates:
## mean of x
##  18.10526
```

```
#paired-sample t-test
t.test(x1,x2,paired = TRUE,conf.level = .95) # paired sample at alpha = .05
```

```
##
##  Paired t-test
##
## data:  x1 and x2
## t = -2.4124, df = 18, p-value = 0.02674
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  -2.0678220 -0.1427044
## sample estimates:
## mean difference
##       -1.105263
```

```
t.test(x1,x2,alternative = "less",paired = TRUE,conf.level = .99)#paired sample one-tai
```

```
##
##  Paired t-test
##
## data:  x1 and x2
## t = -2.4124, df = 18, p-value = 0.01337
## alternative hypothesis: true mean difference is less than 0
## 99 percent confidence interval:
##        -Inf 0.06413553
## sample estimates:
## mean difference
```

```
##       -1.105263
```

```
#independent sample t-test
group<-factor(d72$group)
t.test(x~group,var.equal=TRUE,data=d72)
```

```
##
##  Two Sample t-test
##
## data:  x by group
## t = -0.37576, df = 36, p-value = 0.7093
## alternative hypothesis: true difference in means between group 1 and group 2 is not
## 95 percent confidence interval:
##  -7.070719  4.860193
## sample estimates:
## mean in group 1 mean in group 2
##        43.10526        44.21053
```

```
#equal variance assumption
library(car)
```

```
## Loading required package: carData
```

```
leveneTest(x,group)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group  1  0.0109 0.9173
##       36
```

14 two-way ANOVA without bruceR

注意应该使用 `anova()` 或 `summary.aov()` 对模型进行汇报，不能使用 `summary()` 直接汇报模型。

```
anova_data <- data.frame(read_excel("demo 11.xlsx", sheet = 1))
perform <- anova_data$performance
time <- as.factor(anova_data$time)
present <- as.factor(anova_data$presentation)

anova.lm = lm(perform~time*present,data = anova_data)
summary.aov(anova.lm)
```

```
##              Df Sum Sq Mean Sq F value  Pr(>F)
## time          1  11.25   11.25    3.75 0.07067 .
## present       1  11.25   11.25    3.75 0.07067 .
## time:present  1  31.25   31.25   10.42 0.00526 **
## Residuals    16  48.00    3.00
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(anova.lm)
```

```
## Analysis of Variance Table
##
## Response: perform
##              Df Sum Sq Mean Sq F value   Pr(>F)
## time          1  11.25   11.25   3.750 0.070674 .
## present       1  11.25   11.25   3.750 0.070674 .
## time:present  1  31.25   31.25  10.417 0.005265 **
## Residuals    16  48.00    3.00
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
emms1 <-emmeans::emmeans(anova.lm, ~time|present) #simple main effect of time by differ
summary(emms1)
```

```
## present = 0:
##  time emmean    SE df lower.CL upper.CL
##     0      9 0.775 16     7.36    10.64
```

```
##     1      5 0.775 16     3.36     6.64
##
## present = 1:
##  time emmean    SE df lower.CL upper.CL
##     0      8 0.775 16     6.36     9.64
##     1      9 0.775 16     7.36    10.64
##
## Confidence level used: 0.95
```

```
emms2 <-emmeans::emmeans(anova.lm, ~present|time) #simple main effect of present mode b
summary(emms2)
```

```
## time = 0:
##  present emmean    SE df lower.CL upper.CL
##  0            9 0.775 16     7.36    10.64
##  1            8 0.775 16     6.36     9.64
##
## time = 1:
##  present emmean    SE df lower.CL upper.CL
##  0            5 0.775 16     3.36     6.64
##  1            9 0.775 16     7.36    10.64
##
## Confidence level used: 0.95
```

15 mixed ANOVA

```
mixed_data <- read_excel("demo 12-2.xlsx", sheet = 1)
# dependent variable: time1/time2/time3
# between-group variable: group
# within-group variable: time

library(readxl)
library(ggpubr) # ggboxplot
```

```
library(rstatix) # identify_outliers, get_summary_stats, anova_test, get_anova_table
```

```
##
## Attaching package: 'rstatix'
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
library(dplyr) # %>%
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:car':
##
##     recode
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(bruceR)
```

```
##
## bruceR (version 0.8.9)
## BRoadly Useful Convenient and Efficient R functions
##
## Packages also loaded:
## √ dplyr      √ emmeans       √ ggplot2
## √ tidyr      √ effectsize    √ ggtext
## √ stringr    √ performance   √ cowplot
## √ forcats    √ lmerTest      √ see
```

```
## ✓ data.table
##
## Main functions of `bruceR`:
## cc()             Describe()  TTEST()
## add()            Freq()      MANOVA()
## .mean()          Corr()      EMMEANS()
## set.wd()         Alpha()     PROCESS()
## import()         EFA()       model_summary()
## print_table()    CFA()       lavaan_summary()
##
## https://psychbruce.github.io/bruceR/
##
## These R packages are dependencies of `bruceR` but not installed:
## pacman, lmtest, vars, phia, GGally, GPArotation
## ***** Please Install All Dependencies *****
## install.packages("bruceR", dep=TRUE)
```

```
library(afex)
```

```
## ************
## Welcome to afex. For support visit: http://afex.singmann.science/

## - Functions for ANOVAs: aov_car(), aov_ez(), and aov_4()
## - Methods for calculating p-values with mixed(): 'S', 'KR', 'LRT', and 'PB'
## - 'afex_aov' and 'mixed' objects can be passed to emmeans() for follow-up tests
## - NEWS: emmeans() for ANOVA models now uses model = 'multivariate' as default.
## - Get and set global package options with: afex_options()
## - Set orthogonal sum-to-zero contrasts globally: set_sum_contrasts()
## - For example analyses see: browseVignettes("afex")
## ************

##
## Attaching package: 'afex'

## The following object is masked from 'package:lme4':
##
```

```
##     lmer
```

```
# check missing values
sum(is.na(mixed_data$time1))
```

```
## [1] 0
```

```
sum(is.na(mixed_data$time2))
```

```
## [1] 0
```

```
sum(is.na(mixed_data$time3))
```

```
## [1] 0
```

```
sum(is.na(mixed_data$group))
```

```
## [1] 0
```

```
# check outliers
mixed_data %>%
  group_by(group) %>%
  identify_outliers("time1")
```

```
## # A tibble: 3 x 6
##   group time1 time2 time3 is.outlier is.extreme
##   <dbl> <dbl> <dbl> <dbl> <lgl>      <lgl>
## 1     2     3     1     0 TRUE       TRUE
## 2     2     5     3     2 TRUE       TRUE
## 3     3     6     2     1 TRUE       FALSE
```

```
mixed_data %>%
  group_by(group) %>%
  identify_outliers("time2")
```

```
## # A tibble: 1 x 6
##   group time1 time2 time3 is.outlier is.extreme
##   <dbl> <dbl> <dbl> <dbl> <lgl>      <lgl>
## 1     3     2     6     1 TRUE       FALSE
```

```
mixed_data %>%
  group_by(group) %>%
  identify_outliers("time3")
```

```
## [1] group      time1      time2      time3      is.outlier is.extreme
## <0 rows> (or 0-length row.names)
```

```
# check normality
mixed_data %>%
  group_by(group) %>%
  shapiro_test(time1)
```

```
## # A tibble: 3 x 4
##   group variable statistic     p
##   <dbl> <chr>        <dbl> <dbl>
## 1     1 time1        0.881 0.314
## 2     2 time1        0.883 0.325
## 3     3 time1        0.914 0.492
```

```
mixed_data %>%
  group_by(group) %>%
  shapiro_test(time2)
```

```
## # A tibble: 3 x 4
##   group variable statistic      p
##   <dbl> <chr>        <dbl>  <dbl>
## 1     1 time2        0.881 0.314
## 2     2 time2        0.987 0.967
## 3     3 time2        0.779 0.0537
```

```
mixed_data %>%
  group_by(group) %>%
  shapiro_test(time3)
```

```
## # A tibble: 3 x 4
##   group variable statistic     p
```

```
##   <dbl> <chr>        <dbl> <dbl>
## 1     1 time3        0.961 0.814
## 2     2 time3        0.852 0.201
## 3     3 time3        0.961 0.814
```

```
# Computations
mixed_fit <- MANOVA(mixed_data,
                    dvs = "time1:time3",
                    dvs.pattern = "time(.)",
                    between = "group",
                    within = "time", ###CAUTION###
                    sph.correction = "GG")
```

```
##
## Note:
## dvs="time1:time3" is matched to variables:
## time1, time2, time3
```

```
##
## ====== ANOVA (Mixed Design) ======
##
## Descriptives:
##
##  "group" "time"  Mean    S.D. n
##
##   group1  time1 2.400 (1.673) 5
##   group1  time2 3.800 (0.837) 5
##   group1  time3 6.400 (1.140) 5
##   group2  time1 4.000 (0.707) 5
##   group2  time2 3.000 (1.581) 5
##   group2  time3 1.400 (1.342) 5
##   group3  time1 3.600 (1.517) 5
##   group3  time2 3.200 (1.643) 5
##   group3  time3 1.400 (1.140) 5
##
```

```
## Total sample size: N = 15
##
## ANOVA Table:
## Dependent variable(s):      time1, time2, time3
## Between-subjects factor(s): group
## Within-subjects factor(s):  time
## Covariate(s):                -
##
##                   MS   MSE   df1    df2      F     p      ²p [90% CI of  ²p]   ²G
##
## group         10.289 2.200 2.000 12.000  4.677  .031 *     .438 [.034, .659] .244
## time           0.431 1.877 1.651 19.818  0.229  .755       .019 [.000, .152] .011
## group * time  21.624 1.877 3.303 19.818 11.520 <.001 ***   .658 [.388, .775] .529
##
## Sphericity correction method: GG (Greenhouse-Geisser)
## MSE = mean square error (the residual variance of the linear model)
##  ²p = partial eta-squared = SS / (SS + SSE) = F * df1 / (F * df1 + df2)
##  ²p = partial omega-squared = (F - 1) * df1 / (F * df1 + df2 + 1)
##  ²G = generalized eta-squared (see Olejnik & Algina, 2003)
## Cohen' s f² =  ²p / (1 -  ²p)
##
## Levene' s Test for Homogeneity of Variance:
##
##            Levene' s F df1 df2     p
##
## DV: time1       1.848   2  12  .200
## DV: time2       0.681   2  12  .525
## DV: time3       0.321   2  12  .731
##
##
## Mauchly' s Test of Sphericity:
##
##               Mauchly's W     p
```

```
##
## time               0.7890  .272
## group * time       0.7890  .272
##
```

```
# Post hoc
EMMEANS(mixed_fit, effect = "time", by = "group")
```

```
## ------ EMMEANS (effect = "time") ------
##
## Joint Tests of "time":
##
##  Effect "group" df1 df2      F     p      ²p [90% CI of  ²p]
##
##    time  group1   2  12 18.219 <.001 ***   .752 [.465, .857]
##    time  group2   2  12  7.356  .008 **    .551 [.148, .734]
##    time  group3   2  12  6.941  .010 **    .536 [.131, .725]
##
## Note. Simple effects of repeated measures with 3 or more levels
## are different from the results obtained with SPSS MANOVA syntax.
##
## Estimated Marginal Means of "time":
##
##  "time" "group" Mean [95% CI of Mean]    S.E.
##
##   time1  group1  2.400 [1.069, 3.731] (0.611)
##   time2  group1  3.800 [2.434, 5.166] (0.627)
##   time3  group1  6.400 [5.220, 7.580] (0.542)
##   time1  group2  4.000 [2.669, 5.331] (0.611)
##   time2  group2  3.000 [1.634, 4.366] (0.627)
##   time3  group2  1.400 [0.220, 2.580] (0.542)
##   time1  group3  3.600 [2.269, 4.931] (0.611)
##   time2  group3  3.200 [1.834, 4.566] (0.627)
##   time3  group3  1.400 [0.220, 2.580] (0.542)
```

```
##
##
## Pairwise Comparisons of "time":
##
##       Contrast "group" Estimate    S.E. df      t     p     Cohen' s d [95% CI of d]
##
##  time2 - time1  group1    1.400 (0.917) 12  1.528  .458      0.859 [-0.704,  2.422]
##  time3 - time1  group1    4.000 (0.816) 12  4.899  .001 **   2.454 [ 1.062,  3.846]
##  time3 - time2  group1    2.600 (0.594) 12  4.374  .003 **   1.595 [ 0.581,  2.609]
##  time2 - time1  group2   -1.000 (0.917) 12 -1.091  .890     -0.613 [-2.176,  0.949]
##  time3 - time1  group2   -2.600 (0.816) 12 -3.184  .024 *   -1.595 [-2.987, -0.203]
##  time3 - time2  group2   -1.600 (0.594) 12 -2.692  .059 .   -0.982 [-1.995,  0.032]
##  time2 - time1  group3   -0.400 (0.917) 12 -0.436 1.000     -0.245 [-1.808,  1.317]
##  time3 - time1  group3   -2.200 (0.816) 12 -2.694  .059 .   -1.350 [-2.742,  0.043]
##  time3 - time2  group3   -1.800 (0.594) 12 -3.028  .032 *   -1.104 [-2.118, -0.091]
##
## Pooled SD for computing Cohen' s d: 1.630
## P-value adjustment: Bonferroni method for 3 tests.
##
## Disclaimer:
## By default, pooled SD is Root Mean Square Error (RMSE).
## There is much disagreement on how to compute Cohen' s d.
## You are completely responsible for setting `sd.pooled`.
## You might also use `effectsize::t_to_d()` to compute d.
```

```
EMMEANS(mixed_fit, effect = "group", by = "time")
```

```
## ------ EMMEANS (effect = "group") ------
##
## Joint Tests of "group":
##
##  Effect "time" df1 df2      F     p      ²p [90% CI of  ²p]
##
##   group  time1   2  12  1.857  .198       .236 [.000, .500]
```

```
##   group  time2   2  12  0.441  .654       .068 [.000, .291]
##   group  time3   2  12 28.409 <.001 ***   .826 [.612, .900]
##
## Note. Simple effects of repeated measures with 3 or more levels
## are different from the results obtained with SPSS MANOVA syntax.
##
## Estimated Marginal Means of "group":
##
##  "group" "time" Mean [95% CI of Mean]    S.E.
##
##   group1  time1  2.400 [1.069, 3.731] (0.611)
##   group2  time1  4.000 [2.669, 5.331] (0.611)
##   group3  time1  3.600 [2.269, 4.931] (0.611)
##   group1  time2  3.800 [2.434, 5.166] (0.627)
##   group2  time2  3.000 [1.634, 4.366] (0.627)
##   group3  time2  3.200 [1.834, 4.566] (0.627)
##   group1  time3  6.400 [5.220, 7.580] (0.542)
##   group2  time3  1.400 [0.220, 2.580] (0.542)
##   group3  time3  1.400 [0.220, 2.580] (0.542)
##
##
## Pairwise Comparisons of "group":
##
##         Contrast "time" Estimate    S.E. df      t     p     Cohen' s d [95% CI of d
##
##  group2 - group1  time1    1.600 (0.864) 12  1.852  .266      0.982 [-0.492,  2.455]
##  group3 - group1  time1    1.200 (0.864) 12  1.389  .570      0.736 [-0.737,  2.210]
##  group3 - group2  time1   -0.400 (0.864) 12 -0.463 1.000     -0.245 [-1.719,  1.228]
##  group2 - group1  time2   -0.800 (0.887) 12 -0.902 1.000     -0.491 [-2.003,  1.022]
##  group3 - group1  time2   -0.600 (0.887) 12 -0.676 1.000     -0.368 [-1.880,  1.144]
##  group3 - group2  time2    0.200 (0.887) 12  0.225 1.000      0.123 [-1.390,  1.635]
##  group2 - group1  time3   -5.000 (0.766) 12 -6.528 <.001 *** -3.067 [-4.373, -1.761]
##  group3 - group1  time3   -5.000 (0.766) 12 -6.528 <.001 *** -3.067 [-4.373, -1.761]
```

```
##  group3 - group2  time3   -0.000 (0.766) 12 -0.000 1.000     -0.000 [-1.306,  1.306]
##
## Pooled SD for computing Cohen' s d: 1.630
## P-value adjustment: Bonferroni method for 3 tests.
##
## Disclaimer:
## By default, pooled SD is Root Mean Square Error (RMSE).
## There is much disagreement on how to compute Cohen' s d.
## You are completely responsible for setting `sd.pooled`.
## You might also use `effectsize::t_to_d()` to compute d.
```

```
EMMEANS(mixed_fit, effect = "time")
```

```
## ------ EMMEANS (effect = "time") ------
##
## Joint Tests of "time":
##
##        Effect df1 df2      F     p      ²p [90% CI of  ²p]
##
##  group          2  12  4.677  .031 *     .438 [.034, .659]
##  time           2  12  0.394  .683       .062 [.000, .277]
##  group * time   4  12 16.061 <.001 ***   .843 [.627, .908]
##
## Note. Simple effects of repeated measures with 3 or more levels
## are different from the results obtained with SPSS MANOVA syntax.
##
## Estimated Marginal Means of "time":
##
##  "time" Mean [95% CI of Mean]    S.E.
##
##   time1  3.333 [2.565, 4.102] (0.353)
##   time2  3.333 [2.544, 4.122] (0.362)
##   time3  3.067 [2.385, 3.748] (0.313)
##
```

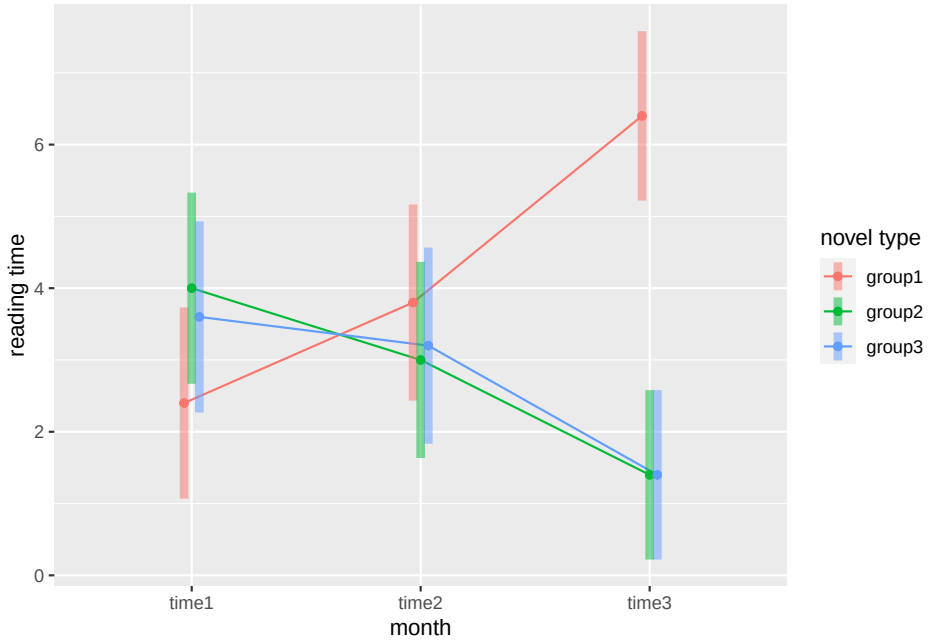
```
##
## Pairwise Comparisons of "time":
##
##       Contrast Estimate    S.E. df      t     p     Cohen' s d [95% CI of d]
##
##  time2 - time1    0.000 (0.529) 12  0.000 1.000       0.000 [-0.902, 0.902]
##  time3 - time1   -0.267 (0.471) 12 -0.566 1.000      -0.164 [-0.967, 0.640]
##  time3 - time2   -0.267 (0.343) 12 -0.777 1.000      -0.164 [-0.749, 0.422]
##
## Pooled SD for computing Cohen' s d: 1.630
## Results are averaged over the levels of: group
## P-value adjustment: Bonferroni method for 3 tests.
##
## Disclaimer:
## By default, pooled SD is Root Mean Square Error (RMSE).
## There is much disagreement on how to compute Cohen' s d.
## You are completely responsible for setting `sd.pooled`.
## You might also use `effectsize::t_to_d()` to compute d.
```

```
EMMEANS(mixed_fit, effect = "group")
```

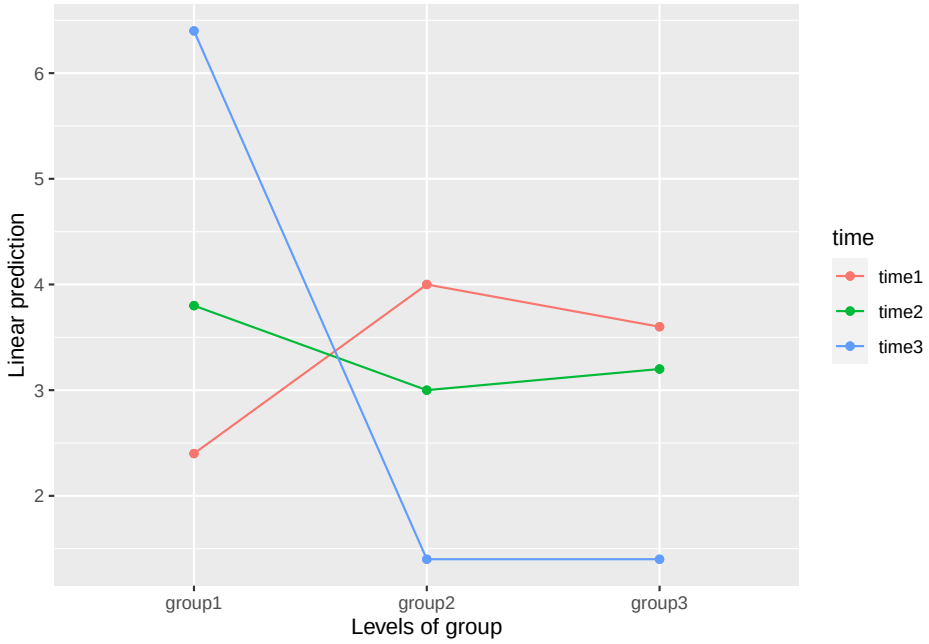
```
## ------ EMMEANS (effect = "group") ------
##
## Joint Tests of "group":
##
##        Effect df1 df2      F     p      ²p [90% CI of  ²p]
##
##  group          2  12  4.677  .031 *     .438 [.034, .659]
##  time           2  12  0.394  .683       .062 [.000, .277]
##  group * time   4  12 16.061 <.001 ***   .843 [.627, .908]
##
## Note. Simple effects of repeated measures with 3 or more levels
## are different from the results obtained with SPSS MANOVA syntax.
##
```

```
## Estimated Marginal Means of "group":
##
##  "group" Mean [95% CI of Mean]    S.E.
##
##   group1  4.200 [3.366, 5.034] (0.383)
##   group2  2.800 [1.966, 3.634] (0.383)
##   group3  2.733 [1.899, 3.568] (0.383)
##
##
## Pairwise Comparisons of "group":
##
##         Contrast Estimate    S.E. df      t     p     Cohen' s d [95% CI of d]
##
##  group2 - group1   -1.400 (0.542) 12 -2.585  .072 .    -0.859 [-1.782, 0.065]
##  group3 - group1   -1.467 (0.542) 12 -2.708  .057 .    -0.900 [-1.823, 0.024]
##  group3 - group2   -0.067 (0.542) 12 -0.123 1.000      -0.041 [-0.964, 0.883]
##
## Pooled SD for computing Cohen' s d: 1.630
## Results are averaged over the levels of: time
## P-value adjustment: Bonferroni method for 3 tests.
##
## Disclaimer:
## By default, pooled SD is Root Mean Square Error (RMSE).
## There is much disagreement on how to compute Cohen' s d.
## You are completely responsible for setting `sd.pooled`.
## You might also use `effectsize::t_to_d()` to compute d.
```

```
# Plotting
emmip(mixed_fit,"group"~"time",CIs=TRUE,xlab="month",ylab="reading time",tlab="novel ty
```




```
emmip(mixed_fit,"time"~"group") #x-axis = novel type, break by time
```



```r
# a memory performance example: measured across 5 days, subjects grouped by two memory
# dependent variable: day1/day2/day3/day4/day5
# between-group variable: group
# within-group variable: time

library(readxl)
library(ggpubr) # ggboxplot
library(rstatix) # identify_outliers, get_summary_stats, anova_test, get_anova_table
library(dplyr) # %>%
library(bruceR)
library(afex)

mixed_data <- read_excel("demo 12-1.xlsx", sheet = 1)

# check missing values
sum(is.na(mixed_data$day1))
```

```
## [1] 0
```

```r
sum(is.na(mixed_data$day2))
```

```
## [1] 0
```

```r
sum(is.na(mixed_data$day3))
```

```
## [1] 0
```

```r
sum(is.na(mixed_data$day4))
```

```
## [1] 0
```

```r
sum(is.na(mixed_data$day5))
```

```
## [1] 0
```

```r
sum(is.na(mixed_data$group))
```

```
## [1] 0
```

```
# check outliers
mixed_data %>%
  group_by(group) %>%
  identify_outliers("day1")
```

```
## [1] group      no         day1       day2       day3       day4       day5
## [8] is.outlier is.extreme
## <0 rows> (or 0-length row.names)
```

```
mixed_data %>%
  group_by(group) %>%
  identify_outliers("day2")
```

```
## # A tibble: 1 x 9
##   group    no  day1  day2  day3  day4  day5 is.outlier is.extreme
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <lgl>      <lgl>
## 1     2    11    45    40    33    25    18 TRUE       FALSE
```

```
mixed_data %>%
  group_by(group) %>%
  identify_outliers("day3")
```

```
## # A tibble: 2 x 9
##   group    no  day1  day2  day3  day4  day5 is.outlier is.extreme
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <lgl>      <lgl>
## 1     2    11    45    40    33    25    18 TRUE       FALSE
## 2     2    12    29    25    17    13     8 TRUE       FALSE
```

```
mixed_data %>%
  group_by(group) %>%
  identify_outliers("day4")
```

```
## [1] group      no         day1       day2       day3       day4       day5
## [8] is.outlier is.extreme
## <0 rows> (or 0-length row.names)
```

```
mixed_data %>%
  group_by(group) %>%
  identify_outliers("day5")
```

```
## [1] group      no         day1       day2       day3       day4       day5
## [8] is.outlier is.extreme
## <0 rows> (or 0-length row.names)
```

```
# check normality
mixed_data %>%
  group_by(group) %>%
  shapiro_test(day1)
```

```
## # A tibble: 2 x 4
##   group variable statistic     p
##   <dbl> <chr>        <dbl> <dbl>
## 1     1 day1         0.947 0.676
## 2     2 day1         0.902 0.304
```

```
mixed_data %>%
  group_by(group) %>%
  shapiro_test(day2)
```

```
## # A tibble: 2 x 4
##   group variable statistic      p
##   <dbl> <chr>        <dbl>  <dbl>
## 1     1 day2         0.834 0.0653
## 2     2 day2         0.924 0.461
```

```
mixed_data %>%
  group_by(group) %>%
  shapiro_test(day3)
```

```
## # A tibble: 2 x 4
##   group variable statistic     p
##   <dbl> <chr>        <dbl> <dbl>
```

```
## 1     1 day3         0.899 0.284
## 2     2 day3         0.968 0.883
```

```
mixed_data %>%
  group_by(group) %>%
  shapiro_test(day4)
```

```
## # A tibble: 2 x 4
##   group variable statistic     p
##   <dbl> <chr>        <dbl> <dbl>
## 1     1 day4         0.968 0.882
## 2     2 day4         0.943 0.636
```

```
mixed_data %>%
  group_by(group) %>%
  shapiro_test(day5)
```

```
## # A tibble: 2 x 4
##   group variable statistic     p
##   <dbl> <chr>        <dbl> <dbl>
## 1     1 day5         0.911 0.363
## 2     2 day5         0.944 0.646
```

```
# Computations
mixed_fit <- MANOVA(mixed_data,
                    dvs = "day1:day5",
                    dvs.pattern = "day(.)",
                    between = "group",
                    within = "time", ###CAUTION###
                    sph.correction = "GG")
```

```
##
## Note:
## dvs="day1:day5" is matched to variables:
## day1, day2, day3, day4, day5

##
```

```
## ====== ANOVA (Mixed Design) ======
##
## Descriptives:
##
##  "group" "time"   Mean    S.D. n
##
##   group1  time1 34.250 (6.228) 8
##   group1  time2 30.875 (6.728) 8
##   group1  time3 24.500 (4.986) 8
##   group1  time4 19.125 (5.592) 8
##   group1  time5 16.875 (5.890) 8
##   group2  time1 35.000 (5.928) 8
##   group2  time2 31.625 (5.097) 8
##   group2  time3 24.875 (4.704) 8
##   group2  time4 20.250 (3.882) 8
##   group2  time5 15.250 (5.651) 8
##
## Total sample size: N = 16
##
## ANOVA Table:
## Dependent variable(s):      day1, day2, day3, day4, day5
## Between-subjects factor(s): group
## Within-subjects factor(s):  time
## Covariate(s):               -
##
##                      MS     MSE   df1    df2       F     p      ²p [90% CI of  ²p]   ²
##
## group            1.513 124.227 1.000 14.000   0.012  .914       .001 [.000, .073] .0
## time          2049.340  15.150 1.870 26.184 135.268 <.001 ***   .906 [.843, .937] .6
## group * time    10.252  15.150 1.870 26.184   0.677  .507       .046 [.000, .194] .0
##
## Sphericity correction method: GG (Greenhouse-Geisser)
## MSE = mean square error (the residual variance of the linear model)
```

```
##  ²p = partial eta-squared = SS / (SS + SSE) = F * df1 / (F * df1 + df2)
##  ²p = partial omega-squared = (F - 1) * df1 / (F * df1 + df2 + 1)
##  ²G = generalized eta-squared (see Olejnik & Algina, 2003)
## Cohen' s f² =  ²p / (1 -  ²p)
##
## Levene' s Test for Homogeneity of Variance:
##
##           Levene' s F df1 df2     p
##
## DV: day1       0.026   1  14  .875
## DV: day2       1.533   1  14  .236
## DV: day3       0.465   1  14  .506
## DV: day4       1.820   1  14  .199
## DV: day5       0.069   1  14  .797
##
##
## Mauchly' s Test of Sphericity:
##
##               Mauchly's W     p
##
## time               0.0738 <.001 ***
## group * time       0.0738 <.001 ***
##
```

```
# post hoc
EMMEANS(mixed_fit, effect = "time", by = "group")
```

```
## ------ EMMEANS (effect = "time") ------
##
## Joint Tests of "time":
##
##  Effect "group" df1 df2      F     p      ²p [90% CI of  ²p]
##
##    time  group1   4  14 30.052 <.001 ***   .896 [.773, .938]
```

```
##    time  group2   4  14 27.325 <.001 ***   .886 [.753, .932]
##
## Note. Simple effects of repeated measures with 3 or more levels
## are different from the results obtained with SPSS MANOVA syntax.
##
## Estimated Marginal Means of "time":
##
##  "time" "group"   Mean [95% CI of Mean]    S.E.
##
##   time1  group1 34.250 [29.640, 38.860] (2.150)
##   time2  group1 30.875 [26.349, 35.401] (2.110)
##   time3  group1 24.500 [20.825, 28.175] (1.714)
##   time4  group1 19.125 [15.475, 22.775] (1.702)
##   time5  group1 16.875 [12.498, 21.252] (2.041)
##   time1  group2 35.000 [30.390, 39.610] (2.150)
##   time2  group2 31.625 [27.099, 36.151] (2.110)
##   time3  group2 24.875 [21.200, 28.550] (1.714)
##   time4  group2 20.250 [16.600, 23.900] (1.702)
##   time5  group2 15.250 [10.873, 19.627] (2.041)
##
##
## Pairwise Comparisons of "time":
##
##       Contrast "group" Estimate    S.E. df       t     p     Cohen' s d [95% CI of d
##
##  time2 - time1  group1   -3.375 (0.962) 14  -3.507  .035 *   -0.928 [-1.808, -0.048]
##  time3 - time1  group1   -9.750 (1.396) 14  -6.983 <.001 *** -2.681 [-3.958, -1.404]
##  time3 - time2  group1   -6.375 (0.923) 14  -6.908 <.001 *** -1.753 [-2.597, -0.909]
##  time4 - time1  group1  -15.125 (1.489) 14 -10.157 <.001 *** -4.159 [-5.521, -2.797]
##  time4 - time2  group1  -11.750 (1.300) 14  -9.036 <.001 *** -3.231 [-4.420, -2.042]
##  time4 - time3  group1   -5.375 (0.754) 14  -7.124 <.001 *** -1.478 [-2.168, -0.788]
##  time5 - time1  group1  -17.375 (1.963) 14  -8.853 <.001 *** -4.778 [-6.573, -2.983]
##  time5 - time2  group1  -14.000 (1.778) 14  -7.876 <.001 *** -3.850 [-5.476, -2.224]
```

```
##  time5 - time3  group1   -7.625 (1.315) 14  -5.797 <.001 *** -2.097 [-3.300, -0.894]
##  time5 - time4  group1   -2.250 (0.874) 14  -2.575  .220     -0.619 [-1.418,  0.180]
##  time2 - time1  group2   -3.375 (0.962) 14  -3.507  .035 *   -0.928 [-1.808, -0.048]
##  time3 - time1  group2  -10.125 (1.396) 14  -7.251 <.001 *** -2.784 [-4.061, -1.507]
##  time3 - time2  group2   -6.750 (0.923) 14  -7.315 <.001 *** -1.856 [-2.700, -1.012]
##  time4 - time1  group2  -14.750 (1.489) 14  -9.905 <.001 *** -4.056 [-5.418, -2.694]
##  time4 - time2  group2  -11.375 (1.300) 14  -8.748 <.001 *** -3.128 [-4.317, -1.939]
##  time4 - time3  group2   -4.625 (0.754) 14  -6.130 <.001 *** -1.272 [-1.962, -0.582]
##  time5 - time1  group2  -19.750 (1.963) 14 -10.063 <.001 *** -5.431 [-7.226, -3.636]
##  time5 - time2  group2  -16.375 (1.778) 14  -9.212 <.001 *** -4.503 [-6.129, -2.877]
##  time5 - time3  group2   -9.625 (1.315) 14  -7.318 <.001 *** -2.647 [-3.850, -1.444]
##  time5 - time4  group2   -5.000 (0.874) 14  -5.723 <.001 *** -1.375 [-2.174, -0.576]
##
## Pooled SD for computing Cohen' s d: 3.636
## P-value adjustment: Bonferroni method for 10 tests.
##
## Disclaimer:
## By default, pooled SD is Root Mean Square Error (RMSE).
## There is much disagreement on how to compute Cohen' s d.
## You are completely responsible for setting `sd.pooled`.
## You might also use `effectsize::t_to_d()` to compute d.
```

```
EMMEANS(mixed_fit, effect = "group", by = "time")
```

```
## ------ EMMEANS (effect = "group") ------
##
## Joint Tests of "group":
##
##  Effect "time" df1 df2     F     p      ²p [90% CI of  ²p]
##
##   group  time1   1  14 0.061  .809       .004 [.000, .164]
##   group  time2   1  14 0.063  .805       .004 [.000, .166]
##   group  time3   1  14 0.024  .879       .002 [.000, .112]
##   group  time4   1  14 0.218  .647       .015 [.000, .231]
```

```
##   group  time5   1  14 0.317  .582       .022 [.000, .253]
##
## Note. Simple effects of repeated measures with 3 or more levels
## are different from the results obtained with SPSS MANOVA syntax.
##
## Estimated Marginal Means of "group":
##
##  "group" "time"   Mean [95% CI of Mean]    S.E.
##
##   group1  time1 34.250 [29.640, 38.860] (2.150)
##   group2  time1 35.000 [30.390, 39.610] (2.150)
##   group1  time2 30.875 [26.349, 35.401] (2.110)
##   group2  time2 31.625 [27.099, 36.151] (2.110)
##   group1  time3 24.500 [20.825, 28.175] (1.714)
##   group2  time3 24.875 [21.200, 28.550] (1.714)
##   group1  time4 19.125 [15.475, 22.775] (1.702)
##   group2  time4 20.250 [16.600, 23.900] (1.702)
##   group1  time5 16.875 [12.498, 21.252] (2.041)
##   group2  time5 15.250 [10.873, 19.627] (2.041)
##
##
## Pairwise Comparisons of "group":
##
##         Contrast "time" Estimate    S.E. df      t     p     Cohen' s d [95% CI of d
##
##  group2 - group1  time1    0.750 (3.040) 14  0.247  .809       0.206 [-1.587, 1.999]
##  group2 - group1  time2    0.750 (2.984) 14  0.251  .805       0.206 [-1.554, 1.966]
##  group2 - group1  time3    0.375 (2.423) 14  0.155  .879       0.103 [-1.326, 1.532]
##  group2 - group1  time4    1.125 (2.407) 14  0.467  .647       0.309 [-1.110, 1.729]
##  group2 - group1  time5   -1.625 (2.886) 14 -0.563  .582      -0.447 [-2.149, 1.255]
##
## Pooled SD for computing Cohen' s d: 3.636
## No need to adjust p values.
```

```
##
## Disclaimer:
## By default, pooled SD is Root Mean Square Error (RMSE).
## There is much disagreement on how to compute Cohen' s d.
## You are completely responsible for setting `sd.pooled`.
## You might also use `effectsize::t_to_d()` to compute d.
```

```
EMMEANS(mixed_fit, effect = "time")
```

```
## ------ EMMEANS (effect = "time") ------
##
## Joint Tests of "time":
##
##        Effect df1 df2      F     p      ²p [90% CI of  ²p]
##
##  group          1  14  0.012  .914       .001 [.000, .072]
##  time           4  14 55.375 <.001 ***   .941 [.871, .965]
##  group * time   4  14  2.002  .149       .364 [.000, .561]
##
## Note. Simple effects of repeated measures with 3 or more levels
## are different from the results obtained with SPSS MANOVA syntax.
##
## Estimated Marginal Means of "time":
##
##  "time"   Mean [95% CI of Mean]    S.E.
##
##   time1 34.625 [31.365, 37.885] (1.520)
##   time2 31.250 [28.050, 34.450] (1.492)
##   time3 24.688 [22.089, 27.286] (1.212)
##   time4 19.688 [17.107, 22.268] (1.203)
##   time5 16.062 [12.968, 19.157] (1.443)
##
##
## Pairwise Comparisons of "time":
```

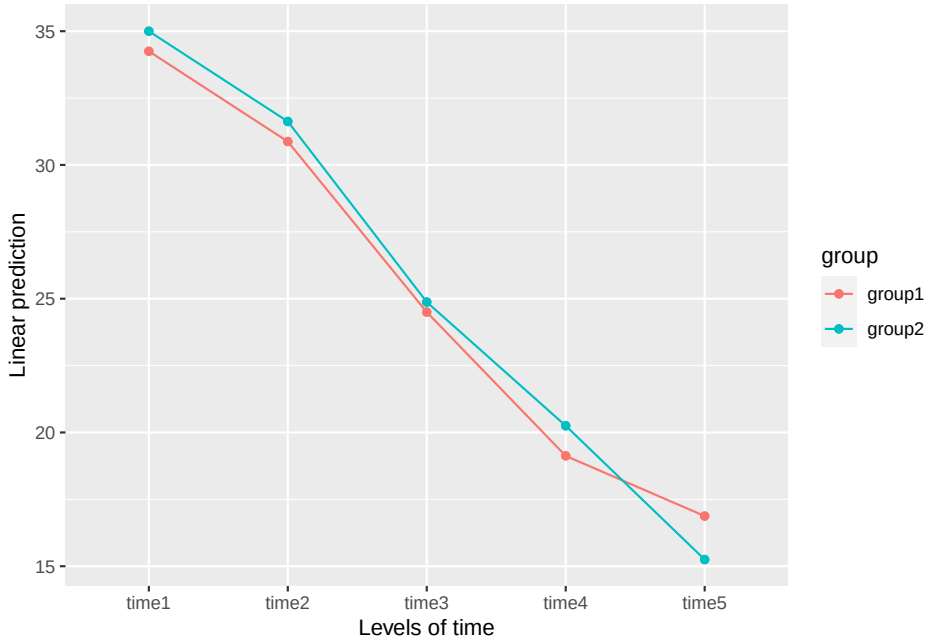
```
##
##       Contrast Estimate    S.E. df       t     p     Cohen' s d [95% CI of d]
##
##  time2 - time1   -3.375 (0.681) 14  -4.959  .002 **  -0.928 [-1.551, -0.306]
##  time3 - time1   -9.938 (0.987) 14 -10.065 <.001 *** -2.733 [-3.636, -1.830]
##  time3 - time2   -6.562 (0.653) 14 -10.057 <.001 *** -1.805 [-2.401, -1.208]
##  time4 - time1  -14.938 (1.053) 14 -14.186 <.001 *** -4.108 [-5.071, -3.145]
##  time4 - time2  -11.562 (0.919) 14 -12.575 <.001 *** -3.180 [-4.021, -2.339]
##  time4 - time3   -5.000 (0.533) 14  -9.372 <.001 *** -1.375 [-1.863, -0.887]
##  time5 - time1  -18.562 (1.388) 14 -13.376 <.001 *** -5.105 [-6.374, -3.835]
##  time5 - time2  -15.188 (1.257) 14 -12.083 <.001 *** -4.177 [-5.326, -3.027]
##  time5 - time3   -8.625 (0.930) 14  -9.274 <.001 *** -2.372 [-3.222, -1.521]
##  time5 - time4   -3.625 (0.618) 14  -5.867 <.001 *** -0.997 [-1.562, -0.432]
##
## Pooled SD for computing Cohen' s d: 3.636
## Results are averaged over the levels of: group
## P-value adjustment: Bonferroni method for 10 tests.
##
## Disclaimer:
## By default, pooled SD is Root Mean Square Error (RMSE).
## There is much disagreement on how to compute Cohen' s d.
## You are completely responsible for setting `sd.pooled`.
## You might also use `effectsize::t_to_d()` to compute d.
```

```
EMMEANS(mixed_fit, effect = "group")
```

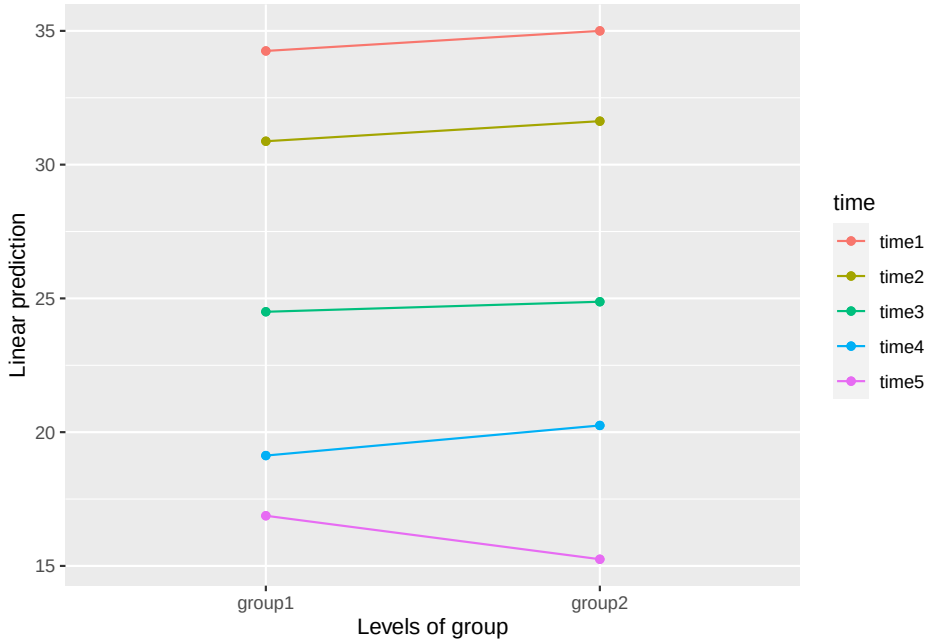
```
## ------ EMMEANS (effect = "group") ------
##
## Joint Tests of "group":
##
##        Effect df1 df2      F     p      ²p [90% CI of  ²p]
##
##  group          1  14  0.012  .914       .001 [.000, .072]
##  time           4  14 55.375 <.001 ***   .941 [.871, .965]
```

```
##  group * time   4  14  2.002  .149       .364 [.000, .561]
##
## Note. Simple effects of repeated measures with 3 or more levels
## are different from the results obtained with SPSS MANOVA syntax.
##
## Estimated Marginal Means of "group":
##
##  "group"   Mean [95% CI of Mean]    S.E.
##
##   group1 25.125 [21.345, 28.905] (1.762)
##   group2 25.400 [21.620, 29.180] (1.762)
##
##
## Pairwise Comparisons of "group":
##
##         Contrast Estimate    S.E. df     t     p     Cohen' s d [95% CI of d]
##
##  group2 - group1    0.275 (2.492) 14 0.110  .914       0.076 [-1.394, 1.546]
##
## Pooled SD for computing Cohen' s d: 3.636
## Results are averaged over the levels of: time
## No need to adjust p values.
##
## Disclaimer:
## By default, pooled SD is Root Mean Square Error (RMSE).
## There is much disagreement on how to compute Cohen' s d.
## You are completely responsible for setting `sd.pooled`.
## You might also use `effectsize::t_to_d()` to compute d.
```

```
emmip(mixed_fit,"group"~"time") #x-axis = time, break by presentation
```




```
emmip(mixed_fit,"time"~"group") #x-axis = present, break by time
```




```
# 相关系数 * cor.test * bruceR::Corr * rstatix::cor_test *
PerformanceAnalytics::chart.correlation
```

```r
# continuous variables = age, salary, salbegin, previous experience
# ordinal variables = education
# nominal variables = gender, minority, job categories

library(haven) # read sav file
library(rstatix)
library(bruceR)

corr_data <- read_sav("demo 13.sav")
age <- corr_data$age
salary <- corr_data$salary
begin_salary <- corr_data$salbegin
experience <- corr_data$prevexp

# check normality
  shapiro_test(age)
```

```
## # A tibble: 1 x 3
##   variable statistic  p.value
##   <chr>        <dbl>    <dbl>
## 1 age          0.869 1.61e-19
```

```r
  shapiro_test(salary)
```

```
## # A tibble: 1 x 3
##   variable statistic  p.value
##   <chr>        <dbl>    <dbl>
## 1 salary       0.771 3.29e-25
```

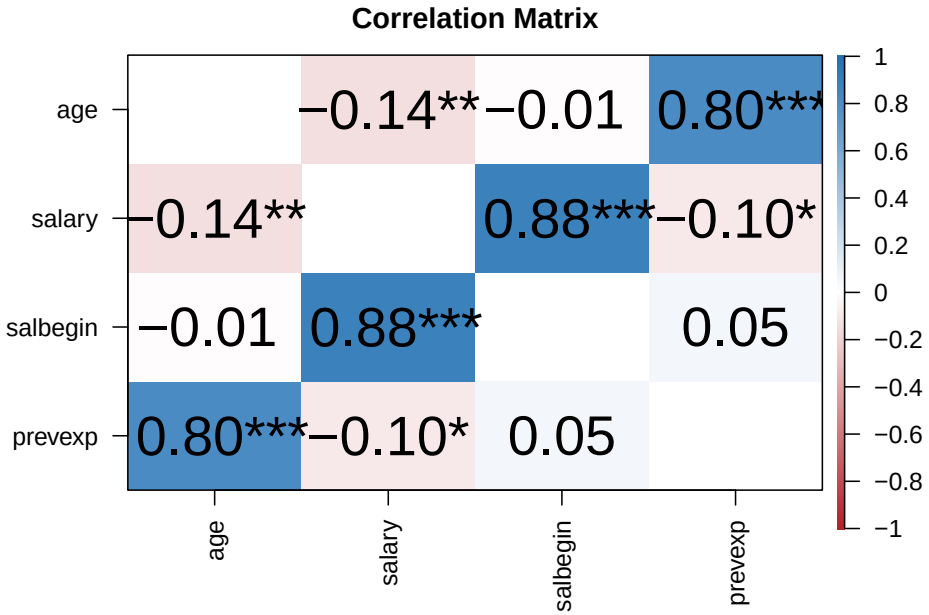
```r
  shapiro_test(begin_salary)
```

```
## # A tibble: 1 x 3
##   variable     statistic  p.value
##   <chr>            <dbl>    <dbl>
## 1 begin_salary     0.715 1.32e-27
```

```
  shapiro_test(experience)
```

```
## # A tibble: 1 x 3
##   variable   statistic  p.value
##   <chr>          <dbl>    <dbl>
## 1 experience     0.814 5.15e-23
```

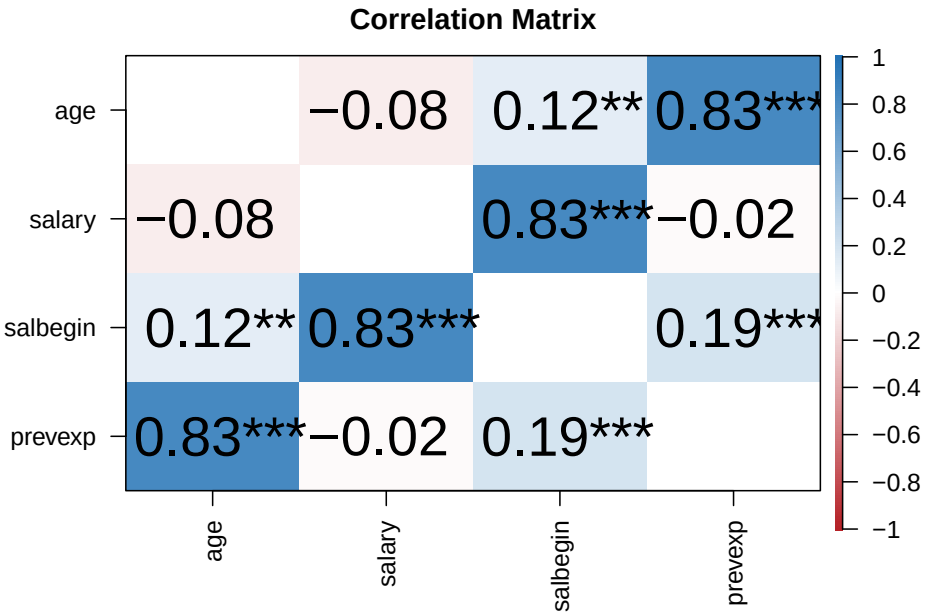
```
# pearson correlation
newdata <- corr_data[,c("age","salary","salbegin","prevexp")]
Corr(newdata)
```




```
## Correlation matrix is displayed in the RStudio `Plots` Pane.
##
## Pearson's r and 95% confidence intervals:
##
##                        r       [95% CI]     p       N
##
## age-salary         -0.14 [-0.23, -0.05]  .002 **  473
## age-salbegin       -0.01 [-0.10,  0.08]  .833     473
## age-prevexp         0.80 [ 0.77,  0.83] <.001 *** 473
```

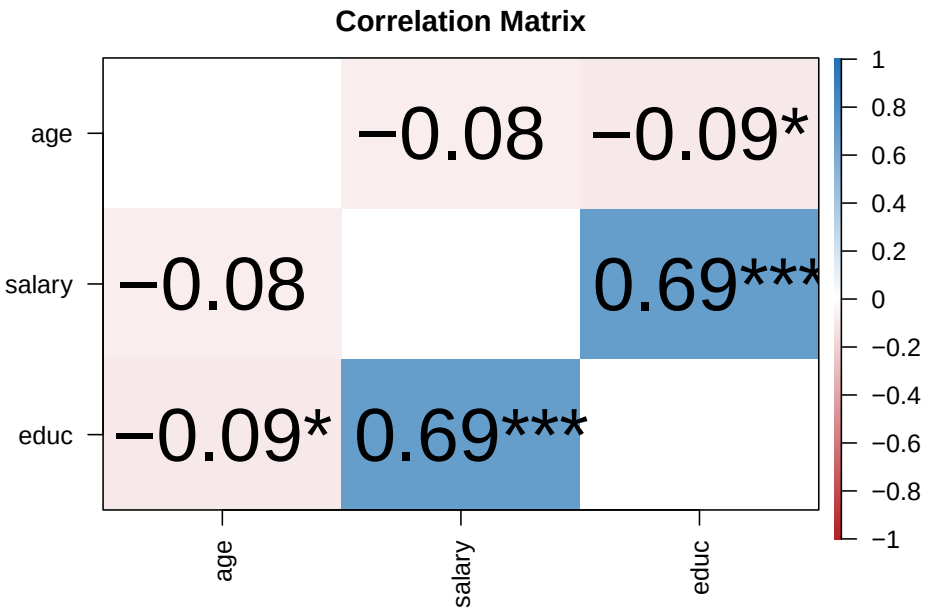
```
## salary-salbegin    0.88 [ 0.86,  0.90] <.001 *** 474
## salary-prevexp    -0.10 [-0.19, -0.01]  .034 *   474
## salbegin-prevexp   0.05 [-0.05,  0.13]  .327     474
##
```

```
Corr(newdata,method = "spearman")
```




```
## Correlation matrix is displayed in the RStudio `Plots` Pane.
##
## Spearman's r and 95% confidence intervals:
##
##                        r      [95% CI]     p       N
##
## age-salary         -0.08 [-0.17, 0.01]  .092 .   473
## age-salbegin        0.12 [ 0.03, 0.21]  .008 **  473
## age-prevexp         0.83 [ 0.79, 0.85] <.001 *** 473
## salary-salbegin     0.83 [ 0.79, 0.85] <.001 *** 474
## salary-prevexp     -0.02 [-0.11, 0.07]  .625     474
## salbegin-prevexp    0.19 [ 0.10, 0.27] <.001 *** 474
##
```

```
# spearman correlation
newdata2 <- corr_data[,c("age","salary","educ")]
Corr(newdata2,method = "spearman")
```




```
## Correlation matrix is displayed in the RStudio `Plots` Pane.
##
## Spearman's r and 95% confidence intervals:
##
##                  r       [95% CI]     p       N
##
## age-salary   -0.08 [-0.17,  0.01]  .092 .   473
## age-educ     -0.09 [-0.18, -0.00]  .040 *   473
## salary-educ   0.69 [ 0.64,  0.73] <.001 *** 474
##
```

```
#visualize your results
library(PerformanceAnalytics)
```

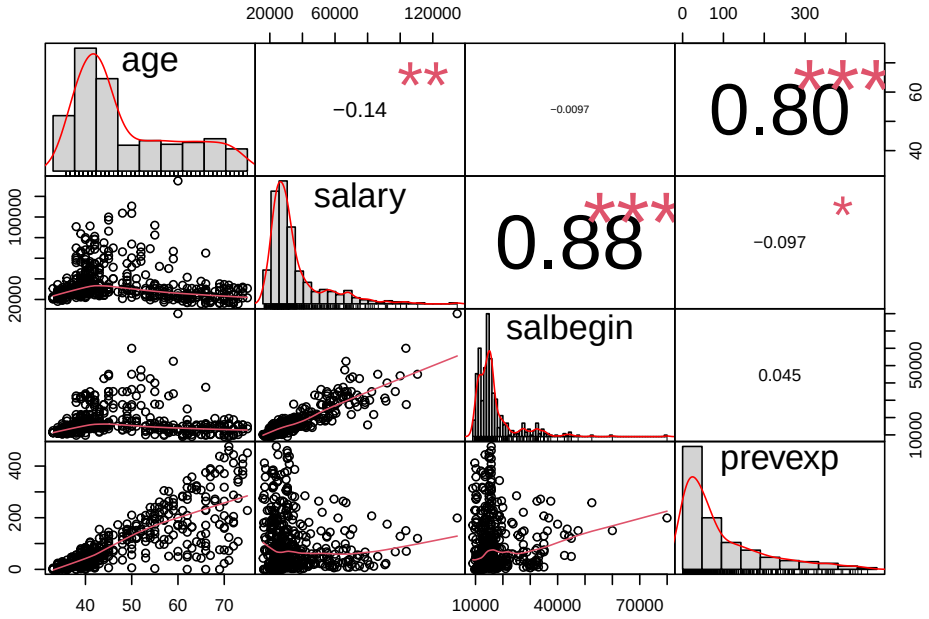
```
## Loading required package: xts
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

##
## Attaching package: 'xts'

## The following objects are masked from 'package:data.table':
##
##     first, last

## The following objects are masked from 'package:dplyr':
##
##     first, last

##
## Attaching package: 'PerformanceAnalytics'

## The following objects are masked from 'package:moments':
##
##     kurtosis, skewness

## The following object is masked from 'package:graphics':
##
##     legend
```

```
chart.Correlation(newdata,histgram = T,color = "black")
```

```
## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter
```

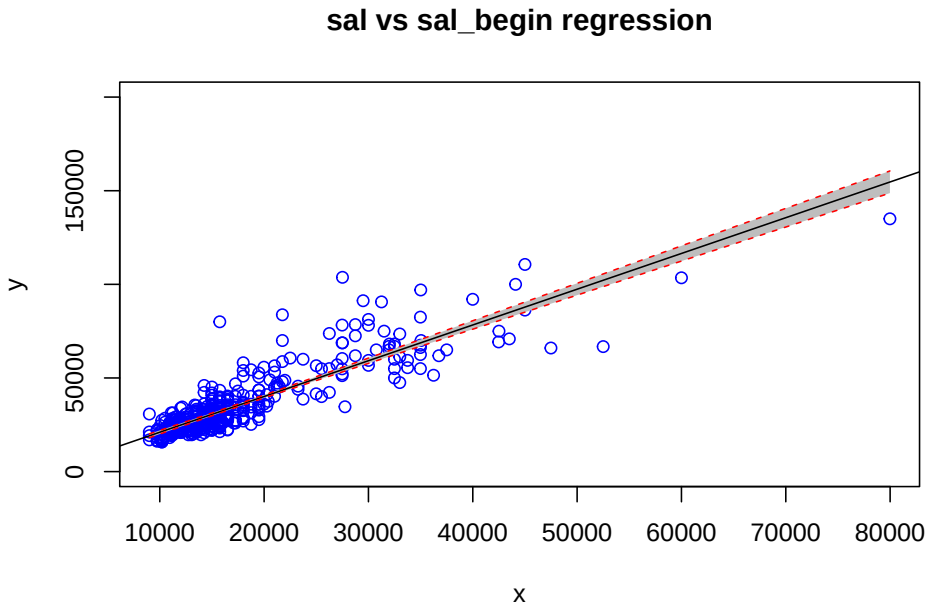
```
## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter
```



线性回归线性回归模型的报告用 summary 或者 summary.lm 均可

```
# salary as a function of beginning salary
reg_data = read_excel("demo 14.xlsx", sheet = 1)
x=reg_data$salbegin
y=reg_data$salary

#plot the data first
plot(x,y,col = "blue",main = "sal vs sal_begin regression",ylim = c(0,200000))
# try to add CI
xrange=seq(min(x),max(x),by=10)
pred=predict(lm(y~x),newdata = data.frame(x=xrange),interval = 'confidence')
polygon(c(xrange,rev(xrange)),c(pred[,2],rev(pred[,3])),col = 'grey',border = NA)
```

```
# add regression line
abline(lm(y~x),cex = 1.3,pch = 16,xlab = "begining salary",ylab = "salary")
# border of intervals
lines(xrange, pred[ ,3], lty = 'dashed', col = 'red')
lines(xrange, pred[ ,2], lty = 'dashed', col = 'red')
```




```
# do the actual regression
lm_results = lm(salary~salbegin,data=reg_data)
summary.lm(lm_results)    # almost all the regression results
```

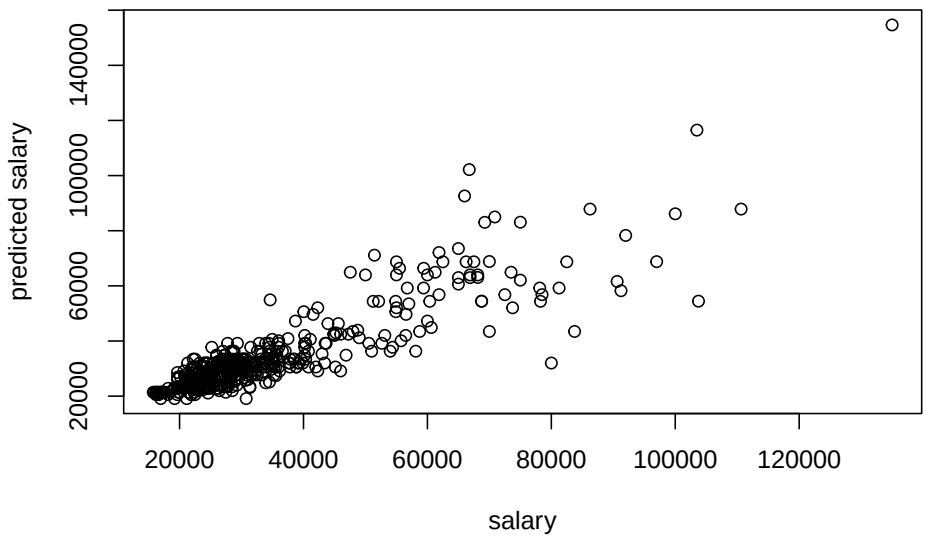
```
##
## Call:
## lm(formula = salary ~ salbegin, data = reg_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -35424  -4031  -1154   2584  49293
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) 1.928e+03  8.887e+02    2.17   0.0305 *
## salbegin    1.909e+00  4.741e-02   40.28   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8115 on 472 degrees of freedom
## Multiple R-squared:  0.7746, Adjusted R-squared:  0.7741
## F-statistic:  1622 on 1 and 472 DF,  p-value: < 2.2e-16
```

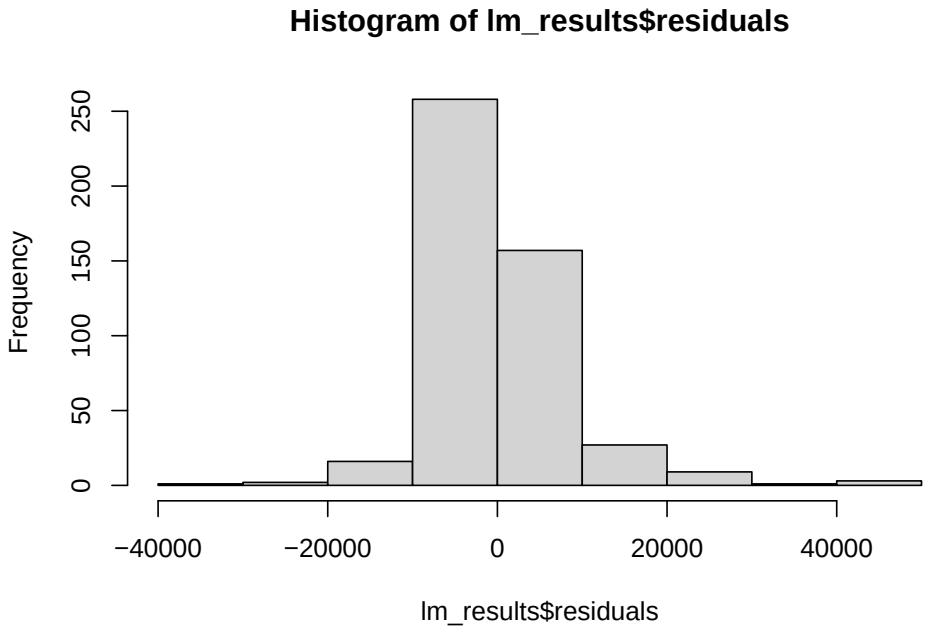
```
summary(lm_results)
```

```
##
## Call:
## lm(formula = salary ~ salbegin, data = reg_data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -35424  -4031  -1154   2584  49293
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.928e+03  8.887e+02    2.17   0.0305 *
## salbegin    1.909e+00  4.741e-02   40.28   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8115 on 472 degrees of freedom
## Multiple R-squared:  0.7746, Adjusted R-squared:  0.7741
## F-statistic:  1622 on 1 and 472 DF,  p-value: < 2.2e-16
```

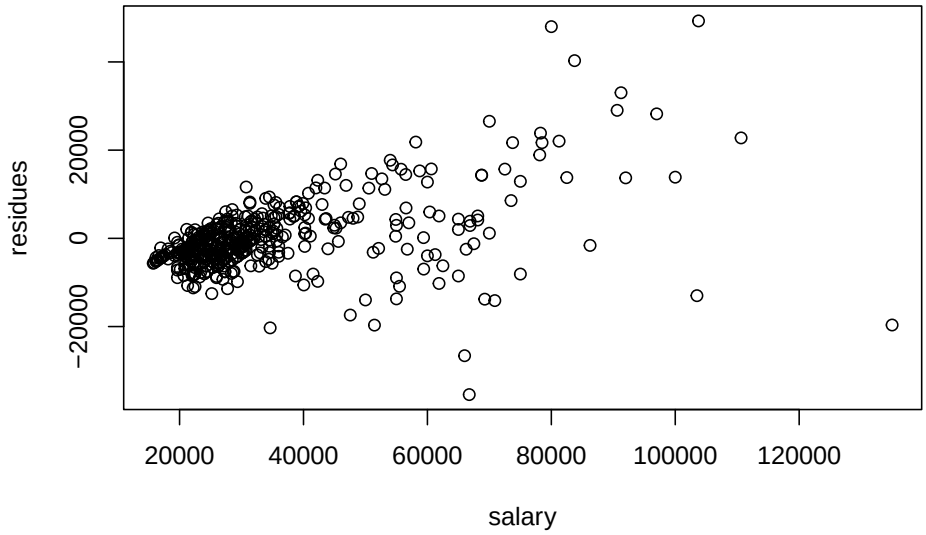
```
#check the prediction vs the actual data
plot(y,lm_results$fitted.values,xlab='salary',ylab='predicted salary')
```



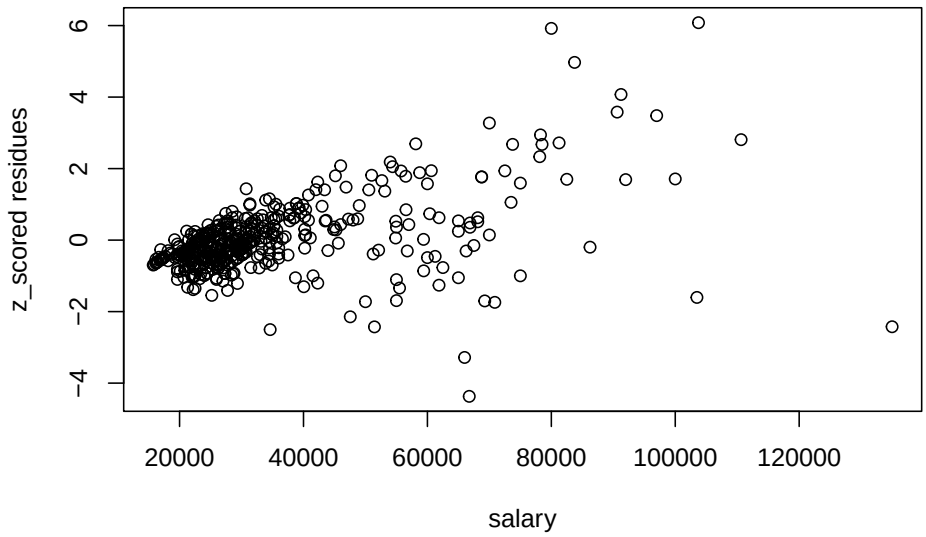

```
# check the residues
hist(lm_results$residuals)
```




```
plot(y,lm_results$residuals,xlab='salary',ylab='residues')
```




```
plot(y,scale(lm_results$residuals),xlab='salary',ylab='z_scored residues')
```




```
## 将残差转化为 z 分数，从而能够更好地判定极端值（可以定为两个 sigma 之外）
```

16 ANOVA

16.1 加载与方差分析相关的包

```
# 前四个与方差分析最紧密相关
library(rstatix) # identify_outliers, and include good stats test functions
library(dplyr) # %>%
library(bruceR) # a good ANOVA related wrapper
library(car) # leveneTest

library(afex) # for factorial design anova
library(ggpubr)  # plot in publication-ready formats
library(purrr)  # for functions dealing with functions
```

```
##
## Attaching package: 'purrr'

## The following object is masked from 'package:data.table':
##
##     transpose

## The following object is masked from 'package:car':
##
##     some
```

16.2 数据初步整理

数据通常有两种存储的形式，一种是长数据，一种是短数据，长数据比较普遍。长数据和短数据的对比就是，长数据还需要根据分组变量将所有的数据分组。

16.2.1 载入数据并且整理为数据框

anova 处理的对象是 Data Frame，因此数据都要使用 `data.frame` 函数转化为数据框。

```
# 这一组数据用于 one-way ANOVA，并且可以用于 repeated-measured ANOVA，strategy 表示自变量
score = c(3,3,4,6,6,8,5,3,5,7,8,8,8,5,8,9,8,10,8,9,7,10,10,10)
subj = rep(1:6,times=4) # 善于使用 rep 函数，创建标签
strategy=c(rep(1,6),rep(2,6),rep(3,6),rep(4,6))
long_data = data.frame(score,subj,strategy) # 构建 dataframe
long_data
```

```
##    score subj strategy
## 1      3    1        1
## 2      3    2        1
## 3      4    3        1
## 4      6    4        1
## 5      6    5        1
## 6      8    6        1
## 7      5    1        2
## 8      3    2        2
## 9      5    3        2
## 10     7    4        2
## 11     8    5        2
## 12     8    6        2
## 13     8    1        3
## 14     5    2        3
## 15     8    3        3
## 16     9    4        3
## 17     8    5        3
## 18    10    6        3
## 19     8    1        4
## 20     9    2        4
## 21     7    3        4
```

```
## 22    10    4         4
## 23    10    5         4
## 24    10    6         4
```

```
# 这一组数据用于 two-way ANOVA，设计的是 2*2 实验，研究 time 和 presentation 对 performanc
perf=c(11,8,9,10,7,4,4,8,5,4,10,10,7,6,7,10,6,10,10,9)
time=c(rep(1,10),rep(0,10))
pre=c(rep(1,5),rep(0,5),rep(1,5),rep(0,5))
id=1:20
two=data.frame(id,time,pre,perf)
two
```

```
##    id time pre perf
## 1   1    1   1   11
## 2   2    1   1    8
## 3   3    1   1    9
## 4   4    1   1   10
## 5   5    1   1    7
## 6   6    1   0    4
## 7   7    1   0    4
## 8   8    1   0    8
## 9   9    1   0    5
## 10 10    1   0    4
## 11 11    0   1   10
## 12 12    0   1   10
## 13 13    0   1    7
## 14 14    0   1    6
## 15 15    0   1    7
## 16 16    0   0   10
## 17 17    0   0    6
## 18 18    0   0   10
## 19 19    0   0   10
## 20 20    0   0    9
```

16.2.2 检验数据本身是否正常

拿到数据以后，应该首先检查是否具有缺失值，是否有极端值（outlier）。

16.2.3 检查缺失值

使用 `is.na` 函数，可以配合 `sum` 看全局。

```
sum(is.na(long_data))
```

```
## [1] 0
```

```
# 可以直接对所有数据检验是否有缺失值，如果没有直接跳过后续步骤

# 具体对每一个列检验缺失值
sum(is.na(long_data$score))
```

```
## [1] 0
```

```
sum(is.na(long_data$subj))
```

```
## [1] 0
```

```
sum(is.na(long_data$strategy))
```

```
## [1] 0
```

```
num_na=long_data %>% map(is.na)
# 也可以使用 map 函数直接对三个列进行检验，但结果比较不直观
summary(num_na)
```

```
##          Length Class  Mode
## score    24     -none- logical
## subj     24     -none- logical
## strategy 24     -none- logical
```

16.2.4 检查极端值

outlier: 值在 $Q_3 + 1.5IQR$ 和 $Q_1 - 1.5IQR$ 区间之外 extreme: 值在 $Q_3 + 3IQR$ 和 $Q_1 - 3IQR$ 区间之外

```
# 检查极端值
long_data %>% # 管道相当于把管道之前的结果传入之后函数的第一个参数
  group_by(strategy) %>% # 注意 group_by 的列并不需要实现转化为 factor 类型
  identify_outliers(score) # 极端值必定是在每组之内讨论的，因此必须先进行分组
```

```
## # A tibble: 2 x 5
##   strategy score  subj is.outlier is.extreme
##      <dbl> <dbl> <int> <lgl>      <lgl>
## 1        3     5     2 TRUE       TRUE
## 2        3    10     6 TRUE       FALSE
```

16.2.5 分组变量转化为因子

数据框中包含的分组变量应该转化为 factor 类型，并且让数据依据其分组。

虽然不是一定要用到，但会比较保险，比如 leveneTest 就需要依照分组变量计算，绘制盒须图等也需要合适的分组变量，更何况，分组变量本身的性质就应该是 factor。

分组变量转为因子类型的方法有：* convert_as_factor：管道友好，可以传入多个参数 * as.factor：简单直白，注意必须是对数据框中的列操作，而不是对原始列的变量操作

```
# 将分组变量转化为因子
# 本例中，subj 和 strategy 都应该转化为因子

# 方法一：使用 convert_as_factor
long_data=data.frame(long_data) %>%
  convert_as_factor(strategy)
# 注意，convert_as_factor 函数是需要接收返回值的，它并不是一种方法，否则数据没有被处理
```

```
long_data # 发现 strategy 转变为了因子
```

```
##    score subj strategy
## 1      3    1        1
## 2      3    2        1
## 3      4    3        1
## 4      6    4        1
## 5      6    5        1
## 6      8    6        1
## 7      5    1        2
## 8      3    2        2
## 9      5    3        2
## 10     7    4        2
## 11     8    5        2
## 12     8    6        2
## 13     8    1        3
## 14     5    2        3
## 15     8    3        3
## 16     9    4        3
## 17     8    5        3
## 18    10    6        3
## 19     8    1        4
## 20     9    2        4
## 21     7    3        4
## 22    10    4        4
## 23    10    5        4
## 24    10    6        4
```

```
# convert_as_factor 可以传入多个参数，可以让所有的要因子化的列都转变为因子类型!

# 方法二: as.factor
# 错误示范，对原始列的变量直接操作
subj=as.factor(subj)
# 改变的只是当时构成 long_data 中的那一列变量，但是 long_data 中的 subj 实际上并没有更新!
```

```
long_data # 发现并没有任何改变发生
```

```
##    score subj strategy
## 1      3    1        1
## 2      3    2        1
## 3      4    3        1
## 4      6    4        1
## 5      6    5        1
## 6      8    6        1
## 7      5    1        2
## 8      3    2        2
## 9      5    3        2
## 10     7    4        2
## 11     8    5        2
## 12     8    6        2
## 13     8    1        3
## 14     5    2        3
## 15     8    3        3
## 16     9    4        3
## 17     8    5        3
## 18    10    6        3
## 19     8    1        4
## 20     9    2        4
## 21     7    3        4
## 22    10    4        4
## 23    10    5        4
## 24    10    6        4
```

```
# 正确示范：修改数据框的列
long_data$subj=as.factor(long_data$subj) # 注意必须是对数据框中的列操作
long_data # 发现 subj 转变为了因子
```

```
##    score subj strategy
## 1      3    1        1
```

```
## 2      3    2        1
## 3      4    3        1
## 4      6    4        1
## 5      6    5        1
## 6      8    6        1
## 7      5    1        2
## 8      3    2        2
## 9      5    3        2
## 10     7    4        2
## 11     8    5        2
## 12     8    6        2
## 13     8    1        3
## 14     5    2        3
## 15     8    3        3
## 16     9    4        3
## 17     8    5        3
## 18    10    6        3
## 19     8    1        4
## 20     9    2        4
## 21     7    3        4
## 22    10    4        4
## 23    10    5        4
## 24    10    6        4
```

综合以上介绍，convert_as_factor 是最通用、最便捷的!

```
long_data=data.frame(long_data) %>%
  convert_as_factor(strategy,subj)
# 注意，convert_as_factor 函数是需要接收返回值的，它并不是一种方法，否则数据没有被处理。
long_data # 发现 strategy 和 subj 都转变为了因子
```

```
##    score subj strategy
## 1      3    1        1
## 2      3    2        1
## 3      4    3        1
```

```
## 4      6    4        1
## 5      6    5        1
## 6      8    6        1
## 7      5    1        2
## 8      3    2        2
## 9      5    3        2
## 10     7    4        2
## 11     8    5        2
## 12     8    6        2
## 13     8    1        3
## 14     5    2        3
## 15     8    3        3
## 16     9    4        3
## 17     8    5        3
## 18    10    6        3
## 19     8    1        4
## 20     9    2        4
## 21     7    3        4
## 22    10    4        4
## 23    10    5        4
## 24    10    6        4
```

```
two=two %>% convert_as_factor(pre,time)
```

16.3 检查 ANOVA 的前提

数据基本无误或排除特殊情况后，根据 anova 的类型，检验需要的前提。比如正态性假设，方差同质假设、球形假设等。

16.3.1 正态性假设的检验

16.3.1.1 管道配合分组 注意这里的正态性检验的函数是 shapiro_test，而不是 shapiro.test

传入数据，使用 group_by 分组（可以同时按多个分组变量实现分组），进行正态性检验 shapiro_test

```
long_data %>%
  group_by(strategy) %>%
  shapiro_test(score)
```

```
## # A tibble: 4 x 4
##   strategy variable statistic     p
##   <fct>    <chr>        <dbl> <dbl>
## 1 1        score        0.896 0.352
## 2 2        score        0.896 0.352
## 3 3        score        0.873 0.238
## 4 4        score        0.831 0.110
```

16.3.1.2 tapply tapply(X, INDEX, FUN = NULL)

- X: 函数的应用对象，在数据分析时，一般是长数据形式
- INDEX: 传入分组变量，有多个分组变量时，用列表承载。每个分组变量都应该与 X 有相同的长度
- FUN: 要应用的函数

tapply 的结果会以列表的形式存储，存储着分组后每一个 cell 中的处理结果，之后用 for 循环遍历即可。

```
normfit=tapply(perf,list(time,pre),shapiro.test)
for (i in 1:2){
  for (j in 1:2){
    print(normfit[i,j])
  }
}
```

```
## [[1]]
##
##  Shapiro-Wilk normality test
##
```

```
## data:  X[[i]]
## W = 0.70079, p-value = 0.009761
##
##
## [[1]]
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.8173, p-value = 0.1113
##
##
## [[1]]
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.70079, p-value = 0.009761
##
##
## [[1]]
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.98676, p-value = 0.9672
```

16.3.2 方差同质假设的检验

- `leveneTest`: 输入公式、输入数据（默认 `center=median`，可以设置为 `center=mean`）
- `levene_test`: 管道友好，输入公式（默认 `center=median`，可以设置为 `center=mean`）

`center=median` 能提供更有鲁棒性的结果，不过一般还是设置为 `center=mean`.

注意，**分组变量需要调整为因子类型**，这一点建议在**数据的基本处理阶段**就要完成！

```
# leveneTest
leveneTest(score~strategy,data=long_data)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group  3  0.9786 0.4226
##       20
```

```
# levene_test (%>%)
long_data %>%
  convert_as_factor(strategy) %>%
  # 如果前期没有将 strategy 转化为因子类型，此处需要加上这条语句
  levene_test(score~strategy)
```

```
## # A tibble: 1 x 4
##     df1   df2 statistic     p
##   <int> <int>     <dbl> <dbl>
## 1     3    20     0.979 0.423
```

```
# 也可以对比发现两种方法最终得到了一样的结果
```

16.3.3 协方差同质假设（球形假设）的检验

`mauchly.test`：第一个参数传入线性回归模型，第二个参数传入公式

```
matrix_data = matrix(long_data$score,nrow=6,ncol=4)
# 首先需要将数据转化为矩阵，其中行表示每个被试（subject），列表示不同的条件（treatment）
mlmfit = lm(matrix_data ~ 1)
# 变量 1~ 变量 2，代表变量 1 根据变量 2 分组，如变量 2 是常数 1，会返回一个拟合的线性模型
mauchly.test(mlmfit,X=~1)
```

```
##
##  Mauchly's test of sphericity
##  Contrasts orthogonal to
##  ~1
##
##
## data:  SSD matrix from lm(formula = matrix_data ~ 1)
## W = 0.55102, p-value = 0.8239
```

```
# 第二个参数指定为 X=~1，代表进行协方差同质性检
# 输出的 p-value 越大越好，认为球形假设成立
```

16.4 ANOVA

- aov,anova 等 R 语言原生函数
- MANOVA：bruceR 包中的强大函数

16.4.1 R 语言原生函数

16.4.1.1 aov aov(formula, data = NULL)

aov 函数可以直接对两列数据进行方差分析，且注意第二列数据应该为因子类型（分组变量）。如果使用数据框，那么 formula 中的变量即为数据框的列名称。

aov 函数返回的是组内和组间的和方、自由度，也即最基本的方差分析的结果。这一结果通常用变量承接，可以认为是一种模型，可以用于后续的分析，比如传入 summary,anova 函数中去。

Residuals 指的是组内。

```
fit_model = aov(score ~ strategy,long_data)
# 前一个为因变量，后一个为分组变量
fit_model
```

```
## Call:
##    aov(formula = score ~ strategy, data = long_data)
##
## Terms:
##                  strategy Residuals
## Sum of Squares         60        62
## Deg. of Freedom         3        20
##
## Residual standard error: 1.760682
## Estimated effects may be unbalanced
```

```
# 要注意，aov 的用法是要分析的数据处于一列，分组数据处于另一列
# 因此，如果有两列数据想要用 aov，需要将两列数据合并，并单独添加一列变量对数据分组

# 例如有 3 组变量，想要通过 aov 拟合：
data1 = 1 : 11
data2 = 5 : 15
data3 = c(10 : 19,21)

# 需要将三个变量合并到一列，并添加分组变量
# 以下代码是最 Raw R 语言的写法

data_y = c(data1, data2, data3)
factor_y = c(
  rep(1, length(data1)),
  rep(2, length(data2)),
  rep(3, length(data3))
)
factor_y = as.factor(factor_y)

aov(data_y ~ factor_y) # 可以直接对现成的两列数据跑 ANOVA
```

```
## Call:
##    aov(formula = data_y ~ factor_y)
```

```
##
## Terms:
##                  factor_y Residuals
## Sum of Squares  456.7273  340.9091
## Deg. of Freedom        2        30
##
## Residual standard error: 3.370999
## Estimated effects may be unbalanced
```

```
# 更规范且常用的方法是将数据整理为数据框
long_test_data=data.frame(group=factor_y,num=data_y)
anova(aov(num~group,long_test_data))
```

```
## Analysis of Variance Table
##
## Response: num
##           Df Sum Sq Mean Sq F value    Pr(>F)
## group      2 456.73 228.364  20.096 2.901e-06 ***
## Residuals 30 340.91  11.364
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# 宽数据可以直接用 MANOVA 处理，迅速将三列 data 合并
fit_data=MANOVA(data.frame(data1,data2,data3),
                dvs='data1:data3',
                dvs.pattern = 'data(.)',
                within  = 'data')
```

```
##
## Note:
## dvs="data1:data3" is matched to variables:
## data1, data2, data3

##
## ====== ANOVA (Within-Subjects Design) ======
##
```

```
## Descriptives:
##
##  "data"   Mean    S.D.  n
##
##   data1  6.000 (3.317) 11
##   data2 10.000 (3.317) 11
##   data3 15.091 (3.477) 11
##
## Total sample size: N = 11
##
## ANOVA Table:
## Dependent variable(s):      data1, data2, data3
## Between-subjects factor(s): -
## Within-subjects factor(s):  data
## Covariate(s):               -
##
##            MS   MSE df1 df2        F     p      ²p [90% CI of  ²p]   ²G
##
## data  228.364 0.030   2  20 7536.000 <.001 ***   .999 [.998, .999] .573
##
## MSE = mean square error (the residual variance of the linear model)
##  ²p = partial eta-squared = SS / (SS + SSE) = F * df1 / (F * df1 + df2)
##  ²p = partial omega-squared = (F - 1) * df1 / (F * df1 + df2 + 1)
##  ²G = generalized eta-squared (see Olejnik & Algina, 2003)
## Cohen' s f² =  ²p / (1 -  ²p)
##
## Levene' s Test for Homogeneity of Variance:
## No between-subjects factors. No need to do the Levene' s test.
##
## Mauchly' s Test of Sphericity:
## The repeated measures have only two levels. The assumption of sphericity is always m
```

```
# 注意的是，宽数据只能进行被试内检验，也即重复测量方差分析。
```

值得一提的是，方差分析是一般线性模型的特例。因此，可以用 aov 模型拟合的数据，理论上都可以用线性模型 (Linear Model) 来拟合。

下方的方差分析结果与上面用 aov 的结果是一致的。

```
lm_model = lm(num~group,long_test_data)
anova(lm_model)
```

```
## Analysis of Variance Table
##
## Response: num
##           Df Sum Sq Mean Sq F value    Pr(>F)
## group      2 456.73 228.364  20.096 2.901e-06 ***
## Residuals 30 340.91  11.364
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

16.4.1.2 anova `anova(object, ...)`

`object`:an object containing the results returned by a model fitting function (e.g., lm or glm).`anova(object, ...)` 中可以包含一个或者多个这样的对象。"When given a single argument it produces a table which tests whether the model terms are significant. When given a sequence of objects, anova tests the models against one another in the order specified."

`anova(object, ...)` 计算的是方差分析表，在原先 `aov` 得到的模型的基础上计算均方、F 值、p 值。

```
anova(fit_model)
```

```
## Analysis of Variance Table
##
## Response: score
##           Df Sum Sq Mean Sq F value   Pr(>F)
## strategy   3     60    20.0  6.4516 0.003116 **
## Residuals 20     62     3.1
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

16.4.1.3 summary

`summary` 也能返回一样的方差分析表。

"The function summary.lm computes and returns a list of summary statistics of the fitted linear model given in object"

```
summary(fit_model)
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## strategy     3     60    20.0   6.452 0.00312 **
## Residuals   20     62     3.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary.aov(fit_model)
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## strategy     3     60    20.0   6.452 0.00312 **
## Residuals   20     62     3.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

16.4.2 bruceR 包中的函数

16.4.2.1 MANOVA (multi-factor ANOVA)

`MANOVA` 只需要提供数据，确定自变量，申明组内和组间的关系三步即可。返回的结果包括方差分析，效应量（gerenral & partial, 90%CI）。

```
MANOVA(data,subID = NULL,dv = NULL,dvs = NULL,dvs.pattern =
NULL,between = NULL,within = NULL,covariate = NULL,ss.type
= "III",sph.correction = "none",aov.include = FALSE,digits =
3,nsmall = digits,file = NULL)
```

- `data`: 可以传入宽数据或长数据

- `subID`：被试的 ID，长数据需要传入，这在长数据中也即列的名称；对于宽数据不需要传入这一参数
- `dv`：自变量
  - 长数据：`dv` 即为因变量
  - 宽数据：`dv` 只有在被试间设计可用，对于被试内或者混合设计，要用 `dvs` 或 `dvs.pattern`
- `dvs`：重复测量方差分析。只适用于宽数据（被试内或混合设计）。传入方式有两种
  - `start:stop`：声明变量的范围
  - charactor vector：向量中包含变量的名称
- `dvs.pattern`：如果使用了 `dvs.`，那么要传入正则表达式明确变量名的样式
- `between`：between-subjects factor(s)，如有多个分组变量，将它们包含在向量中
- `within`：within-subjects factor(s)，如有多个分组变量，将它们包含在向量中
- `sph.correction`：违背球形假设的纠正。只适用于多余三个水平的重复测量方差分析。通常设置为 `GG`

其中，`between` 和 `within` 需要至少明确一个量！如果需要进行重复测量（消除被试个体差异），就用 `within` 指定；否则用 `between` 指定。长数据和宽数据的比较：可以认为长数据包含了组间的信息（自变量），也包含了组内的信息（被试个体）。长数据的优点是简单，可以承载任意维度的信息，但宽数据承载的信息有限，可以把向 `MANOVA` 传参的过程理解为将长数据转变为宽数据的过程。长数据需要明确 `subID`，相当于依据其将数据重新整合，把被试 ID 作为首列；长数据还要明确 `within` 变量，程序再根据 `within` 变量将数据拆分成多列。这样之后长数据变得和宽数据”等价”了。此外，系统需要明白每一行或每一列数据的意义并转化，因此**每一列都必须被指明含义**。

长数据

```
# one-way
repeat_fit <- MANOVA(long_data,
                     subID = "subj", # 明确这一列指代的是被试编号
```

```
                    dv = "score", # 因变量
                    within = "strategy",
                    # strategy 作为 within-subject 的变量，说明进行重复测量方差检验
                    sph.correction = "GG")  # greenhouse-geisser correction
```

```
##
##     * Data are aggregated to mean (across items/trials)
##     if there are >=2 observations per subject and cell.
##     You may use Linear Mixed Model to analyze the data,
##     e.g., with subjects and items as level-2 clusters.

##
## ====== ANOVA (Within-Subjects Design) ======
##
## Descriptives:
##
##  "strategy"  Mean    S.D. n
##
##   strategy1 5.000 (2.000) 6
##   strategy2 6.000 (2.000) 6
##   strategy3 8.000 (1.673) 6
##   strategy4 9.000 (1.265) 6
##
## Total sample size: N = 6
##
## ANOVA Table:
## Dependent variable(s):      score
## Between-subjects factor(s): -
## Within-subjects factor(s):  strategy
## Covariate(s):               -
##
##              MS   MSE   df1    df2      F     p      ²p [90% CI of  ²p]   ²G
##
## strategy  27.551 1.286 2.178 10.889 21.429 <.001 ***   .811 [.558, .894] .492
```

```
##
## Sphericity correction method: GG (Greenhouse-Geisser)
## MSE = mean square error (the residual variance of the linear model)
##  ²p = partial eta-squared = SS / (SS + SSE) = F * df1 / (F * df1 + df2)
##  ²p = partial omega-squared = (F - 1) * df1 / (F * df1 + df2 + 1)
##  ²G = generalized eta-squared (see Olejnik & Algina, 2003)
## Cohen' s f² =  ²p / (1 -  ²p)
##
## Levene' s Test for Homogeneity of Variance:
## No between-subjects factors. No need to do the Levene' s test.
##
## Mauchly' s Test of Sphericity:
##
##           Mauchly's W     p
##
## strategy       0.5510  .824
##
```

```
# two-way
two_fit=MANOVA(two,dv='perf',between = c('time','pre'))
```

```
##
## ====== ANOVA (Between-Subjects Design) ======
##
## Descriptives:
##
##  "time" "pre"  Mean    S.D. n
##
##   time0  pre0 9.000 (1.732) 5
##   time0  pre1 8.000 (1.871) 5
##   time1  pre0 5.000 (1.732) 5
##   time1  pre1 9.000 (1.581) 5
##
## Total sample size: N = 20
```

```
##
## ANOVA Table:
## Dependent variable(s):      perf
## Between-subjects factor(s): time, pre
## Within-subjects factor(s):  -
## Covariate(s):               -
##
##                 MS   MSE df1 df2      F     p      ²p [90% CI of  ²p]   ²G
##
## time        11.250 3.000   1  16  3.750  .071 .     .190 [.000, .454] .190
## pre         11.250 3.000   1  16  3.750  .071 .     .190 [.000, .454] .190
## time * pre  31.250 3.000   1  16 10.417  .005 **    .394 [.095, .617] .394
##
## MSE = mean square error (the residual variance of the linear model)
##  ²p = partial eta-squared = SS / (SS + SSE) = F * df1 / (F * df1 + df2)
##  ²p = partial omega-squared = (F - 1) * df1 / (F * df1 + df2 + 1)
##  ²G = generalized eta-squared (see Olejnik & Algina, 2003)
## Cohen' s f² =  ²p / (1 -  ²p)
##
## Levene' s Test for Homogeneity of Variance:
##
##           Levene' s F df1 df2     p
##
## DV: perf       0.235   3  16  .870
##
```

```
# 因为不做 repeated-measured ANOVA, 所以 subID 可以不用明确是哪一列
```

宽数据

```
wide_data = data.frame(s1 = c(3,3,4,6,6,8),
                       s2 = c(5,3,5,7,8,8),
                       s3=c(8,5,8,9,8,10),
                       s4 =c(8,9,7,10,10,10))
repeated_fit=MANOVA(wide_data,
```

```
                    dvs = 's1:s4',
                    dvs.pattern = 's(.)',
                    within='s',
                    sph.correction = 'GG')
```

```
##
## Note:
## dvs="s1:s4" is matched to variables:
## s1, s2, s3, s4
##
## ====== ANOVA (Within-Subjects Design) ======
##
## Descriptives:
##
##  "s"  Mean    S.D. n
##
##   s1 5.000 (2.000) 6
##   s2 6.000 (2.000) 6
##   s3 8.000 (1.673) 6
##   s4 9.000 (1.265) 6
##
## Total sample size: N = 6
##
## ANOVA Table:
## Dependent variable(s):      s1, s2, s3, s4
## Between-subjects factor(s):  -
## Within-subjects factor(s):  s
## Covariate(s):                -
##
##        MS   MSE   df1    df2      F     p      ²p [90% CI of  ²p]   ²G
##
## s  27.551 1.286 2.178 10.889 21.429 <.001 ***   .811 [.558, .894] .492
##
```

```
## Sphericity correction method: GG (Greenhouse-Geisser)
## MSE = mean square error (the residual variance of the linear model)
##  ²p = partial eta-squared = SS / (SS + SSE) = F * df1 / (F * df1 + df2)
##  ²p = partial omega-squared = (F - 1) * df1 / (F * df1 + df2 + 1)
##  ²G = generalized eta-squared (see Olejnik & Algina, 2003)
## Cohen' s f² =  ²p / (1 -  ²p)
##
## Levene' s Test for Homogeneity of Variance:
## No between-subjects factors. No need to do the Levene' s test.
##
## Mauchly' s Test of Sphericity:
##
##    Mauchly's W     p
##
## s       0.5510  .824
##
```

between-subject design

尝试将上述的 `long_data` 中的 `subj` 一列去除，这相当于不知道我们三种 `strategy` 下用的是同样的一批被试，因此我们只能进行被试间检验 (one-way ANOVA)

```
between_data=data.frame(score,strategy) %>% convert_as_factor(strategy)
# 注：MANOVA 传参时，between 变量可以不是因子类型，照样可以得到一样的结果
between_fit=MANOVA(between_data,
                   dv='score',
                   between='strategy')
```

```
##
## ====== ANOVA (Between-Subjects Design) ======
##
## Descriptives:
##
##  "strategy"  Mean    S.D. n
```

```
##
##   strategy1 5.000 (2.000) 6
##   strategy2 6.000 (2.000) 6
##   strategy3 8.000 (1.673) 6
##   strategy4 9.000 (1.265) 6
##
## Total sample size: N = 24
##
## ANOVA Table:
## Dependent variable(s):      score
## Between-subjects factor(s): strategy
## Within-subjects factor(s):   -
## Covariate(s):                -
##
##               MS   MSE df1 df2     F     p      ²p [90% CI of  ²p]   ²G
##
## strategy  20.000 3.100   3  20 6.452  .003 **    .492 [.164, .656] .492
##
## MSE = mean square error (the residual variance of the linear model)
##  ²p = partial eta-squared = SS / (SS + SSE) = F * df1 / (F * df1 + df2)
##  ²p = partial omega-squared = (F - 1) * df1 / (F * df1 + df2 + 1)
##  ²G = generalized eta-squared (see Olejnik & Algina, 2003)
## Cohen' s f² =  ²p / (1 -  ²p)
##
## Levene' s Test for Homogeneity of Variance:
##
##            Levene' s F df1 df2     p
##
## DV: score       1.067   3  20  .385
##
```

将被试间方差分析（one-way ANOVA）和被试内方差分析（repeated-measured ANOVA) 的结果相互比较，发现后者更显著。有意思的是，当用单因素方差分析时，得到的效应量是 η^2，而使用重复测量方差分析时，得

到的效应量应为 *partial* η^2，这在两种检验的结果都得到了验证。

16.5 事后检验

如果 ANOVA 的结果显著，那么需要紧接着进行事后检验，判定是哪两组的差异显著。事后检验较为保守，即使 ANOVA 显著，事后检验也未必得到显著的结果。

16.5.1 EMMEANS

`EMMEANS(model,effect=NULL,p.adjust='bonferroni')`

`EMMEANS` 能完成两项重要的任务：* 事后检验 * 简单主效应分析

此外，`EMMEANS` 还能够返回效应量 (partial η^2,Cohen's d，及其置信区间)

向 `EMMEANS` 中传入**由 `MANOVA` 得到的方差分析模型**，设置待检验的效应（`effect` 和 `by`），以及调整方式即可。

调整方式默认为 `p.adjust='bonferroni'`，也可以设置为'tukey','scheffe'。

```
# p.adjust='tukey'
EMMEANS(between_fit,effect = 'strategy',p.adjust = 'tukey')
```

16.5.1.0.1 EMMEANS 用于事后检验

```
## ------ EMMEANS (effect = "strategy") ------
##
## Joint Tests of "strategy":
##
##    Effect df1 df2     F     p      ²p [90% CI of  ²p]
##
##  strategy   3  20 6.452  .003 **    .492 [.164, .656]
##
```

```
## Note. Simple effects of repeated measures with 3 or more levels
## are different from the results obtained with SPSS MANOVA syntax.
##
## Estimated Marginal Means of "strategy":
##
##  "strategy" Mean [95% CI of Mean]    S.E.
##
##   strategy1 5.000 [3.501,  6.499] (0.719)
##   strategy2 6.000 [4.501,  7.499] (0.719)
##   strategy3 8.000 [6.501,  9.499] (0.719)
##   strategy4 9.000 [7.501, 10.499] (0.719)
##
##
## Pairwise Comparisons of "strategy":
##
##               Contrast Estimate    S.E. df     t     p     Cohen' s d [95% CI of d]
##
##  strategy2 - strategy1    1.000 (1.017) 20 0.984  .760       0.568 [-1.048, 2.184]
##  strategy3 - strategy1    3.000 (1.017) 20 2.951  .036 *     1.704 [ 0.088, 3.320]
##  strategy3 - strategy2    2.000 (1.017) 20 1.967  .233       1.136 [-0.480, 2.752]
##  strategy4 - strategy1    4.000 (1.017) 20 3.935  .004 **    2.272 [ 0.656, 3.888]
##  strategy4 - strategy2    3.000 (1.017) 20 2.951  .036 *     1.704 [ 0.088, 3.320]
##  strategy4 - strategy3    1.000 (1.017) 20 0.984  .760       0.568 [-1.048, 2.184]
##
## Pooled SD for computing Cohen' s d: 1.761
## P-value adjustment: Tukey method for comparing a family of 4 estimates.
##
## Disclaimer:
## By default, pooled SD is Root Mean Square Error (RMSE).
## There is much disagreement on how to compute Cohen' s d.
## You are completely responsible for setting `sd.pooled`.
## You might also use `effectsize::t_to_d()` to compute d.
```

也可以调整 `p.adjust='scheffe'`，结果会更难显著。

```
EMMEANS(between_fit,effect = 'strategy',p.adjust = 'scheffe')
```

```
## ------ EMMEANS (effect = "strategy") ------
##
## Joint Tests of "strategy":
##
##    Effect df1 df2     F     p      ²p [90% CI of  ²p]
##
##  strategy   3  20 6.452  .003 **    .492 [.164, .656]
##
## Note. Simple effects of repeated measures with 3 or more levels
## are different from the results obtained with SPSS MANOVA syntax.
##
## Estimated Marginal Means of "strategy":
##
##  "strategy" Mean [95% CI of Mean]    S.E.
##
##   strategy1 5.000 [3.501,  6.499] (0.719)
##   strategy2 6.000 [4.501,  7.499] (0.719)
##   strategy3 8.000 [6.501,  9.499] (0.719)
##   strategy4 9.000 [7.501, 10.499] (0.719)
##
##
## Pairwise Comparisons of "strategy":
##
##               Contrast Estimate    S.E. df     t     p     Cohen' s d [95% CI of d]
##
##  strategy2 - strategy1    1.000 (1.017) 20 0.984  .809       0.568 [-1.192, 2.328]
##  strategy3 - strategy1    3.000 (1.017) 20 2.951  .060 .     1.704 [-0.056, 3.464]
##  strategy3 - strategy2    2.000 (1.017) 20 1.967  .305       1.136 [-0.624, 2.896]
##  strategy4 - strategy1    4.000 (1.017) 20 3.935  .008 **    2.272 [ 0.512, 4.032]
##  strategy4 - strategy2    3.000 (1.017) 20 2.951  .060 .     1.704 [-0.056, 3.464]
##  strategy4 - strategy3    1.000 (1.017) 20 0.984  .809       0.568 [-1.192, 2.328]
```

```
##
## Pooled SD for computing Cohen' s d: 1.761
## P-value adjustment: Scheffe method with rank 3.
##
## Disclaimer:
## By default, pooled SD is Root Mean Square Error (RMSE).
## There is much disagreement on how to compute Cohen' s d.
## You are completely responsible for setting `sd.pooled`.
## You might also use `effectsize::t_to_d()` to compute d.
```

16.5.1.0.2 EMMEANS 用于简单主效应分析

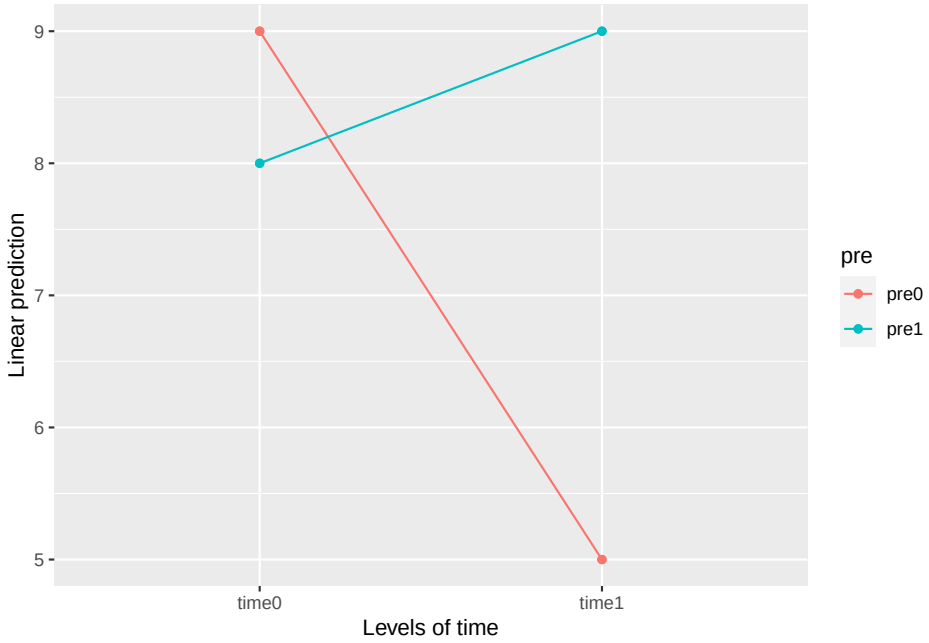
`effect` 参数传入的即为要检验的简单主效应，`by` 参数传入的是分组。

```
EMMEANS(two_fit,effect = 'pre',by = 'time') # 检验 pre 的简单主效应
```

```
## ------ EMMEANS (effect = "pre") ------
##
## Joint Tests of "pre":
##
##  Effect "time" df1 df2      F     p      ²p [90% CI of  ²p]
##
##     pre  time0   1  16  0.833  .375       .049 [.000, .291]
##     pre  time1   1  16 13.333  .002 **    .455 [.146, .659]
##
## Note. Simple effects of repeated measures with 3 or more levels
## are different from the results obtained with SPSS MANOVA syntax.
##
## Estimated Marginal Means of "pre":
##
##  "pre" "time" Mean [95% CI of Mean]    S.E.
##
##   pre0  time0 9.000 [7.358, 10.642] (0.775)
##   pre1  time0 8.000 [6.358,  9.642] (0.775)
##   pre0  time1 5.000 [3.358,  6.642] (0.775)
```

```
##   pre1  time1 9.000 [7.358, 10.642] (0.775)
##
##
## Pairwise Comparisons of "pre":
##
##     Contrast "time" Estimate    S.E. df      t     p     Cohen' s d [95% CI of d]
##
##  pre1 - pre0  time0   -1.000 (1.095) 16 -0.913  .375      -0.577 [-1.918, 0.763]
##  pre1 - pre0  time1    4.000 (1.095) 16  3.651  .002 **    2.309 [ 0.969, 3.650]
##
## Pooled SD for computing Cohen' s d: 1.732
## No need to adjust p values.
##
## Disclaimer:
## By default, pooled SD is Root Mean Square Error (RMSE).
## There is much disagreement on how to compute Cohen' s d.
## You are completely responsible for setting `sd.pooled`.
## You might also use `effectsize::t_to_d()` to compute d.
```

```
emmip(two_fit,pre~time) # 可以绘制 pre 关于 time 分组的简单主效应，其中纵轴为 pre，横轴为
```

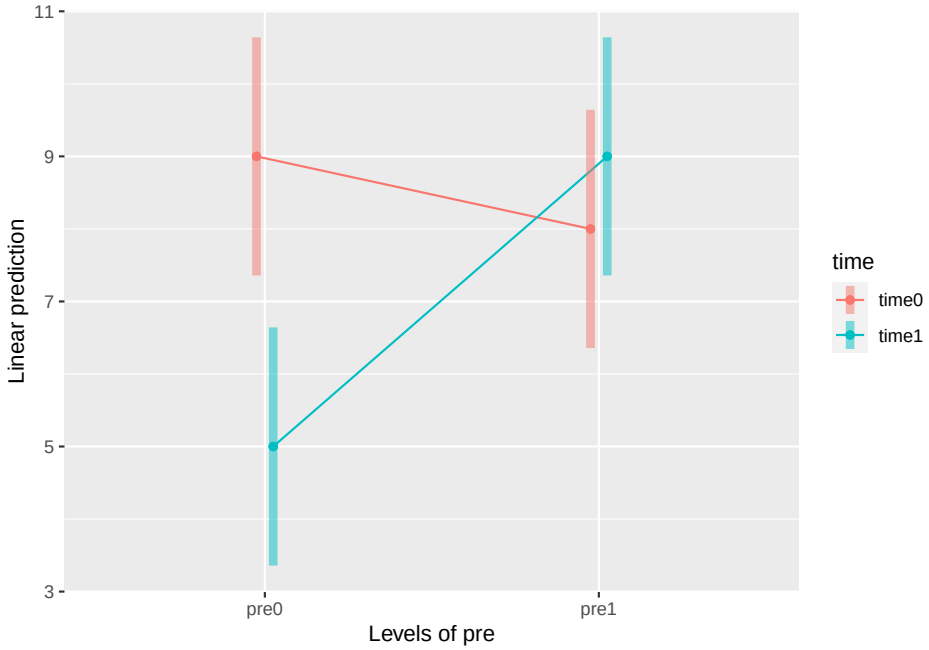



```
EMMEANS(two_fit, effect = "time", by = "pre")
```

```
## ------ EMMEANS (effect = "time") ------
##
## Joint Tests of "time":
##
##  Effect "pre" df1 df2      F     p      ²p [90% CI of  ²p]
##
##    time  pre0   1  16 13.333  .002 **    .455 [.146, .659]
##    time  pre1   1  16  0.833  .375       .049 [.000, .291]
##
## Note. Simple effects of repeated measures with 3 or more levels
## are different from the results obtained with SPSS MANOVA syntax.
##
## Estimated Marginal Means of "time":
##
##  "time" "pre" Mean [95% CI of Mean]    S.E.
##
```

```
##   time0  pre0 9.000 [7.358, 10.642] (0.775)
##   time1  pre0 5.000 [3.358,  6.642] (0.775)
##   time0  pre1 8.000 [6.358,  9.642] (0.775)
##   time1  pre1 9.000 [7.358, 10.642] (0.775)
##
##
## Pairwise Comparisons of "time":
##
##       Contrast "pre" Estimate    S.E. df      t     p     Cohen' s d [95% CI of d]
##
##  time1 - time0  pre0   -4.000 (1.095) 16 -3.651  .002 **  -2.309 [-3.650, -0.969]
##  time1 - time0  pre1    1.000 (1.095) 16  0.913  .375      0.577 [-0.763,  1.918]
##
## Pooled SD for computing Cohen' s d: 1.732
## No need to adjust p values.
##
## Disclaimer:
## By default, pooled SD is Root Mean Square Error (RMSE).
## There is much disagreement on how to compute Cohen' s d.
## You are completely responsible for setting `sd.pooled`.
## You might also use `effectsize::t_to_d()` to compute d.
```

```
emmip(two_fit,time~pre,CIs = T)
```




```
EMMEANS(two_fit,effect = 'pre')
```

```
## ------ EMMEANS (effect = "pre") ------
##
## Joint Tests of "pre":
##
##      Effect df1 df2      F     p      ²p [90% CI of  ²p]
##
##  time         1  16  3.750  .071 .     .190 [.000, .454]
##  pre          1  16  3.750  .071 .     .190 [.000, .454]
##  time * pre   1  16 10.417  .005 **    .394 [.095, .617]
##
## Note. Simple effects of repeated measures with 3 or more levels
## are different from the results obtained with SPSS MANOVA syntax.
##
## Estimated Marginal Means of "pre":
##
##  "pre" Mean [95% CI of Mean]    S.E.
```

```
##
##   pre0  7.000 [5.839, 8.161] (0.548)
##   pre1  8.500 [7.339, 9.661] (0.548)
##
##
## Pairwise Comparisons of "pre":
##
##     Contrast Estimate    S.E. df     t     p     Cohen' s d [95% CI of d]
##
##  pre1 - pre0    1.500 (0.775) 16 1.936  .071 .     0.866 [-0.082, 1.814]
##
## Pooled SD for computing Cohen' s d: 1.732
## Results are averaged over the levels of: time
## No need to adjust p values.
##
## Disclaimer:
## By default, pooled SD is Root Mean Square Error (RMSE).
## There is much disagreement on how to compute Cohen' s d.
## You are completely responsible for setting `sd.pooled`.
## You might also use `effectsize::t_to_d()` to compute d.
```

16.5.2 TukeyHSD

`TukeyHSD(x, conf.level = 0.95, ...)`, 来自原生 `stats` 包。

其中 `x` 是拟合模型，这里通常适用 `aov` 拟合的模型。`conf.level` 可以设定置信区间的大小，通常和显著性水平相对应。

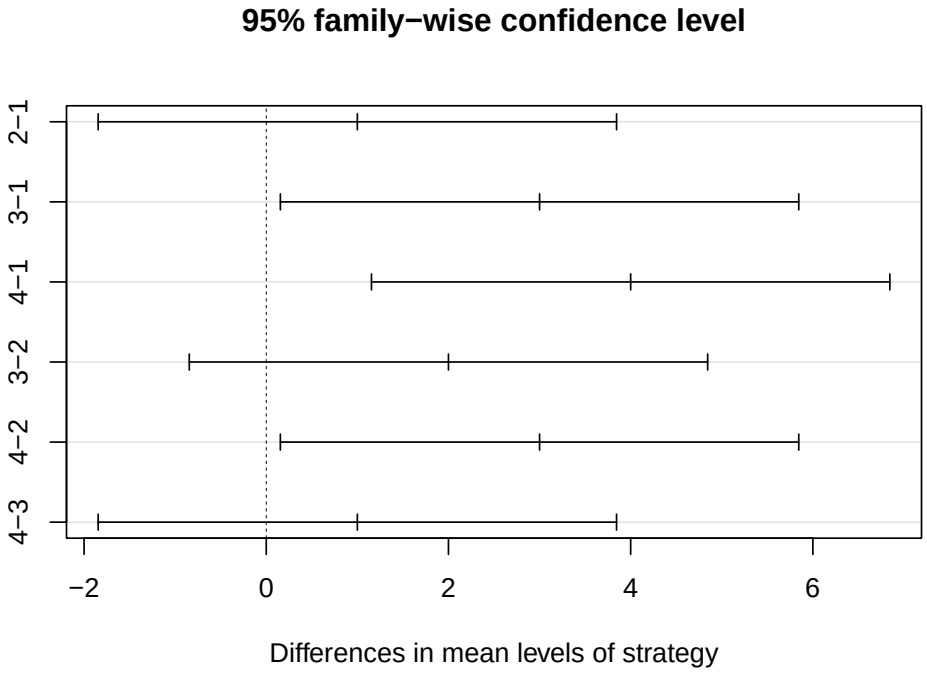
`diff` 是两组的均值差异，`p adj` 是调整后的 p-value。

`TukeyHSD` 类可以用 `print` 显示结果，也可以用 `plot` 展现事后两两比较的图，图上呈现了均值差异和置信区间，非常直观。

```
tfm=TukeyHSD(fit_model)
print(tfm)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = score ~ strategy, data = long_data)
##
## $strategy
##     diff        lwr      upr     p adj
## 2-1    1 -1.8452027 3.845203 0.7601658
## 3-1    3  0.1547973 5.845203 0.0364863
## 4-1    4  1.1547973 6.845203 0.0041906
## 3-2    2 -0.8452027 4.845203 0.2331646
## 4-2    3  0.1547973 5.845203 0.0364863
## 4-3    1 -1.8452027 3.845203 0.7601658
```

```
plot(tfm)
```



16.5.3 tukey_hsd

`tukey_hsd(x, formula, ...)`, 来自 `rstatix` 包。

`tukey_hsd(x, formula, ...)` 是管道友好的。

其中 `x` 为 `aov`,`lm`,`data.frame` 等包含着 `formula` 中的变量的数据类型，`formula` 为 `num~group`，其中 `num` 为数值变量，`group` 为分类变量。如果传入了方差分析模型 `aov`，那么就不用再输入 `formula`，因为 `formula` 在 `aov` 中就已经体现。

`tukey_hsd(x, formula, ...)` 能返回一个 tibble data frame，反馈所有的结果（呈现效果更美观）。

```
# 直接传入 aov 模型
tukey_hsd(fit_model)
```

```
## # A tibble: 6 x 9
##   term     group1 group2 null.value estimate conf.low conf.high   p.adj p.adj.~1
## * <chr>    <chr>  <chr>       <dbl>    <dbl>    <dbl>     <dbl>   <dbl> <chr>
## 1 strategy 1      2               0     1.00   -1.85       3.85 0.76    ns
## 2 strategy 1      3               0     3       0.155      5.85 0.0365  *
## 3 strategy 1      4               0     4       1.15       6.85 0.00419 **
## 4 strategy 2      3               0     2      -0.845      4.85 0.233   ns
## 5 strategy 2      4               0     3       0.155      5.85 0.0365  *
## 6 strategy 3      4               0     1      -1.85       3.85 0.76    ns
## # ... with abbreviated variable name 1: p.adj.signif
```

```
# 使用管道
long_data %>% tukey_hsd(score~strategy)
```

```
## # A tibble: 6 x 9
##   term     group1 group2 null.value estimate conf.low conf.high   p.adj p.adj.~1
## * <chr>    <chr>  <chr>       <dbl>    <dbl>    <dbl>     <dbl>   <dbl> <chr>
## 1 strategy 1      2               0     1.00   -1.85       3.85 0.76    ns
## 2 strategy 1      3               0     3       0.155      5.85 0.0365  *
## 3 strategy 1      4               0     4       1.15       6.85 0.00419 **
```

```
## 4 strategy 2      3                0      2      -0.845       4.85 0.233   ns
## 5 strategy 2      4                0      3       0.155       5.85 0.0365  *
## 6 strategy 3      4                0      1      -1.85        3.85 0.76    ns
## # ... with abbreviated variable name 1: p.adj.signif
```

16.6 补充

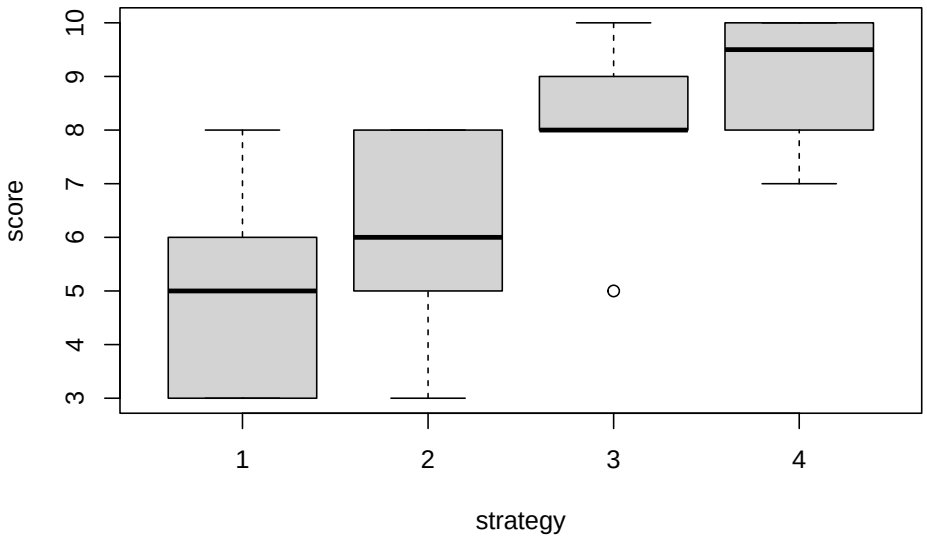
在进行方差分析之前，可以先

1. 绘制箱型图大概比较一下各组数据。纵轴上可以比较均值，宽度可以大致看出方差是否同质。
2. 绘制 QQ 图大致查看正态性假设是否成立

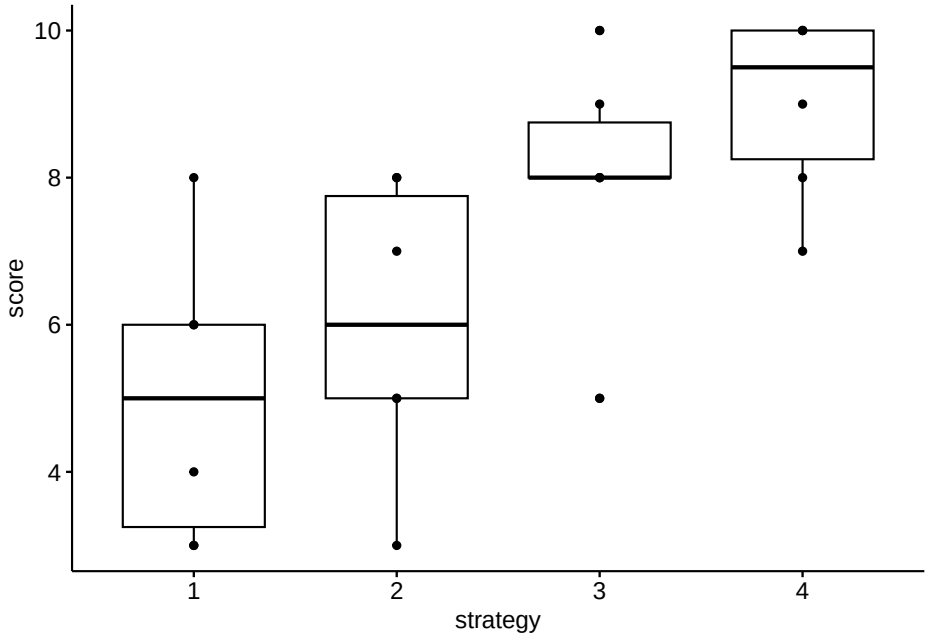
16.6.1 绘制箱型图

- boxplot
- ggboxplot

```
# 传统方法
boxplot(score~strategy,long_data,varwidth=T)
```



```
# 设置 varwidth=T，可以由箱型图的宽度看出各组数据的方差大致情况
# 如果宽度相近，也反映了方差可能同质。

# ggboxplot，画出来的效果可能更美观
ggboxplot(long_data, x = "strategy", y = "score", add = "point")
```



16.6.2 绘制 QQ 图

要根据不同的组别画出 QQ plot，因此使用 `ggqqplot` 函数

`ggqqplot` 函数：第一个参数传入数据框，第二个参数传入要绘制的变量，`facet.by` 传入的是分组变量

```
ggqqplot(long_data, "score", facet.by = "strategy")
```

```
## Warning: The following aesthetics were dropped during statistical transformation: sa
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
```

```
##   variable into a factor?
## The following aesthetics were dropped during statistical transformation: sample
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
## The following aesthetics were dropped during statistical transformation: sample
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
## The following aesthetics were dropped during statistical transformation: sample
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
## The following aesthetics were dropped during statistical transformation: sample
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
## The following aesthetics were dropped during statistical transformation: sample
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
## The following aesthetics were dropped during statistical transformation: sample
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
## The following aesthetics were dropped during statistical transformation: sample
## i This can happen when ggplot fails to infer the correct grouping structure in
```

```
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```