Cheatsheet

Leo

2020/12/27

```
library("tidyverse")
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.0     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## Warning: package 'tibble' was built under R version 4.0.4

## Warning: package 'tidyr' was built under R version 4.0.4

## Warning: package 'dplyr' was built under R version 4.0.4

## Warning: package 'forcats' was built under R version 4.0.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library("ggpubr")
# rm(list = ls())
```

Read files(txt)

header: column name row.names: which column to be the row names ?^space

```
# bp <- read.table ("blood_pressure.txt",header = T,row.names = 1, sep='\t')
```

Get vector

continuous vector

```
a <- seq(1,20,1)
a
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```r
b <- seq(1,20,0.5)
b
```

```
##  [1]  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5  6.0  6.5  7.0  7.5  8.0
## [16]  8.5  9.0  9.5 10.0 10.5 11.0 11.5 12.0 12.5 13.0 13.5 14.0 14.5 15.0 15.5
## [31] 16.0 16.5 17.0 17.5 18.0 18.5 19.0 19.5 20.0
```

replicate

```r
c <- replicate(10,mean(rnorm(10)))
c
```

```
##  [1] -0.56258069 -0.42971760  0.32115199  0.09273652  0.10012562  0.08796602
##  [7] -0.45215031 -0.18440508  0.21596631  0.33255610
```

```r
d <- rep(1,10)
d
```

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

date frame

```r
trial <- read.csv("drug_trial.csv")
summary(trial)
```

```
##   treatment             pain
##  Length:44          Min.   :-1.6485
##  Class :character   1st Qu.: 0.5069
##  Mode  :character   Median : 0.9995
##                     Mean   : 1.0065
##                     3rd Qu.: 1.5582
##                     Max.   : 2.5704
```

```r
nrow(trial)
```

```
## [1] 44
```

```r
ncol(trial)
```

```
## [1] 2
```

```r
trail <- trial[-c(4),] # remove 4th column
unique(trial$treatment) # a list of unique value
```

```
## [1] "placebo" "drugA"   "drugB"
```

```r
sample(trial,2,replace=FALSE) # sample for columns
```

```
##    treatment        pain
## 1    placebo  2.35493558
## 2    placebo  1.38270464
## 3    placebo  0.71267611
## 4    placebo  0.40831384
## 5    placebo  1.69831708
## 6    placebo  1.41274251
## 7    placebo  2.28024592
## 8    placebo -0.20054178
## 9    placebo  1.96714256
## 10   placebo  2.57041769
## 11   placebo  2.30697609
## 12   placebo  0.90166190
## 13   placebo  0.93660510
## 14   placebo  1.93873063
## 15   placebo  0.91111230
## 16     drugA  0.41230740
## 17     drugA  1.56798788
## 18     drugA  1.24273074
## 19     drugA  1.07840196
## 20     drugA  1.55094706
## 21     drugA  1.06230197
## 22     drugA  2.52645539
## 23     drugA  1.13096568
## 24     drugA  1.44379791
## 25     drugA  1.79646637
## 26     drugA  0.53843290
## 27     drugA  1.55498307
## 28     drugA  0.32647057
## 29     drugA  1.74414345
## 30     drugA  1.38454621
## 31     drugB  0.69245923
## 32     drugB  0.19226536
## 33     drugB -1.64851266
## 34     drugB  0.07668351
## 35     drugB -0.27024559
## 36     drugB -0.32047823
## 37     drugB  0.91721163
## 38     drugB  0.89274028
## 39     drugB  0.65363774
## 40     drugB -0.24564002
## 41     drugB  0.64731693
## 42     drugB  0.62837703
## 43     drugB -0.15360754
## 44     drugB  1.28126479
```

```r
sample_n(trial,2,replace=FALSE) # sample for rows
```

```
##   treatment      pain
## 1     drugA 1.7441434
## 2     drugA 0.3264706
```

```r
# Create data frame
data1 <- data.frame(x1=c(1,3,5,7),x2=c(2,4,6,8),x3=c(11,12,13,14),x4=c(15,16,17,18))
data1
```

```
##   x1 x2 x3 x4
## 1  1  2 11 15
## 2  3  4 12 16
## 3  5  6 13 17
## 4  7  8 14 18
```

```r
# add a row
row <- c(1,1,1,1)
data2 <- rbind(data1[1:2,], row, data1[3:nrow(data1),]) # row bind
data2
```

```
##    x1 x2 x3 x4
## 1   1  2 11 15
## 2   3  4 12 16
## 3   1  1  1  1
## 31  5  6 13 17
## 4   7  8 14 18
```

```r
# add a column
y <- 1:4
data2 <- cbind(data1[,1:2],y,data1[,3:ncol(data1)]) # column bind
data2
```

```
##   x1 x2 y x3 x4
## 1  1  2 1 11 15
## 2  3  4 2 12 16
## 3  5  6 3 13 17
## 4  7  8 4 14 18
```

```r
# column names
names(data2)
```

```
## [1] "x1" "x2" "y"  "x3" "x4"
```

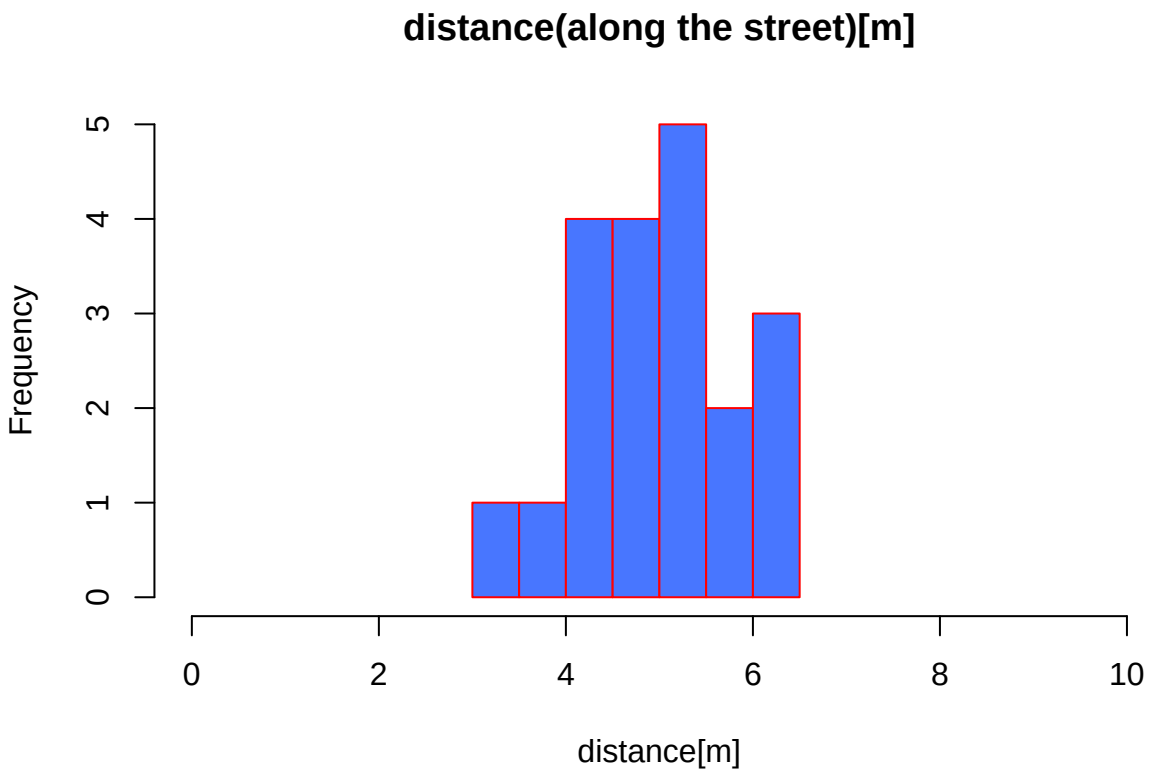
```r
# row names
row.names(data2)
```

```
## [1] "1" "2" "3" "4"
```

Some useful tools to plot the data

looking at the distribution, use histogram

```r
n = rnorm(20,5)
# breaks divides data in n groups
hist(n,main = "distance(along the street)[m]",xlab = "distance[m]",
     xlim = c(0,10),breaks=10,border = "red",col = "royalblue1")
```



if two vectors, combine them into a data frame with index.

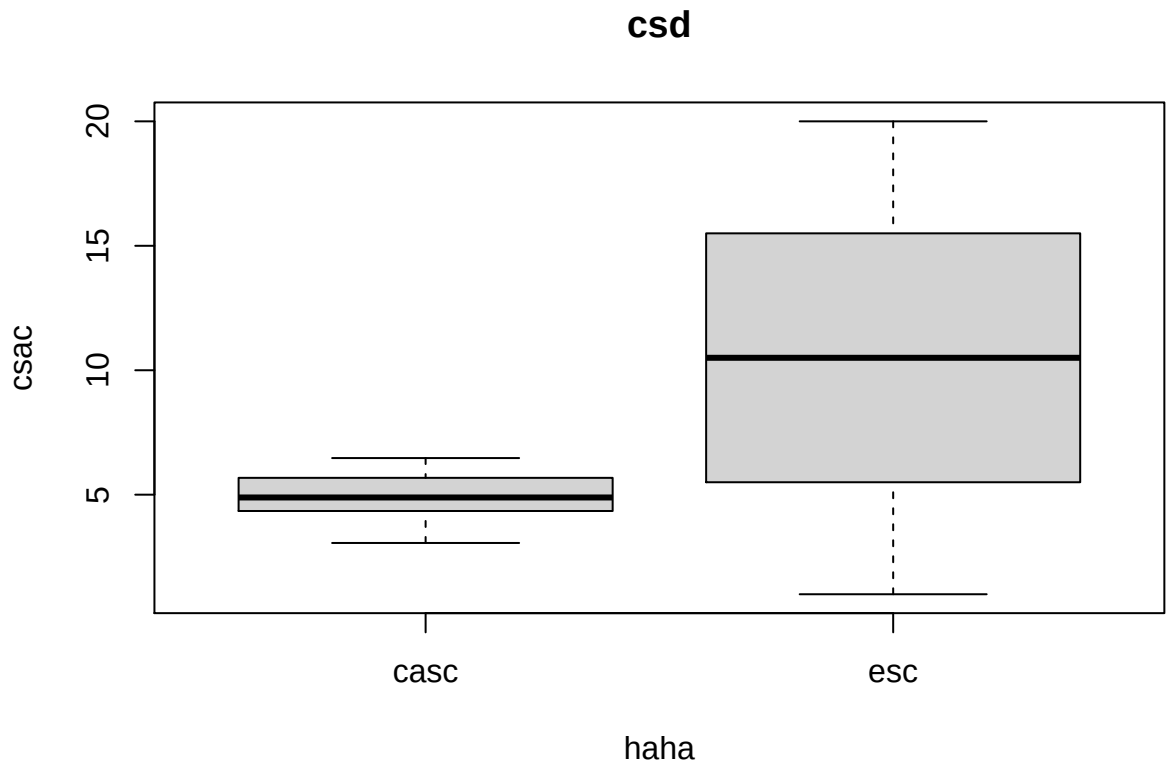
```
response <- c(a,n)
index <- c(rep("esc",20),rep("casc",20))
dataset <- data.frame(response,index)
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 4.0.4
```

```
kable(dataset[1:5,])
```

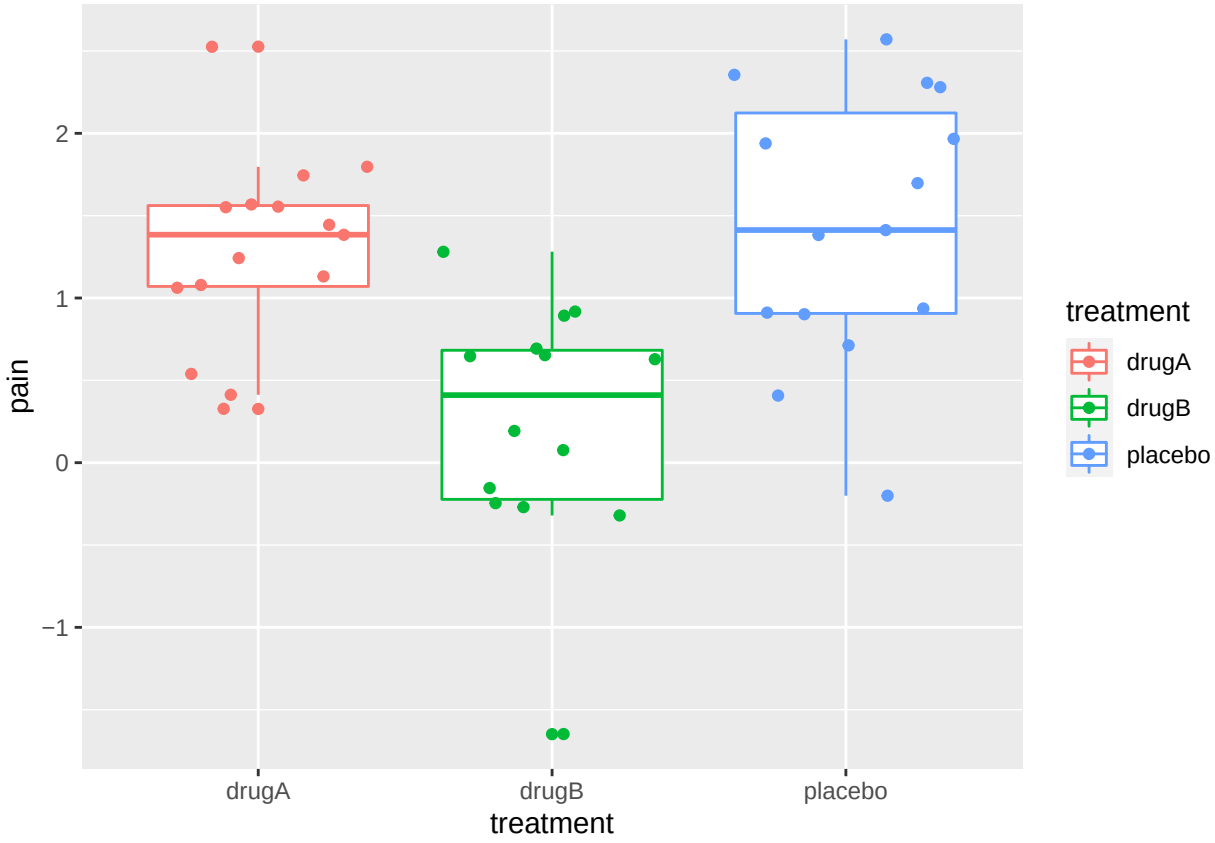
response	index
1	esc
2	esc
3	esc
4	esc
5	esc

```
boxplot( response ~ index, dataset,xlab = "haha", main = "csd",ylab = "csac")
```

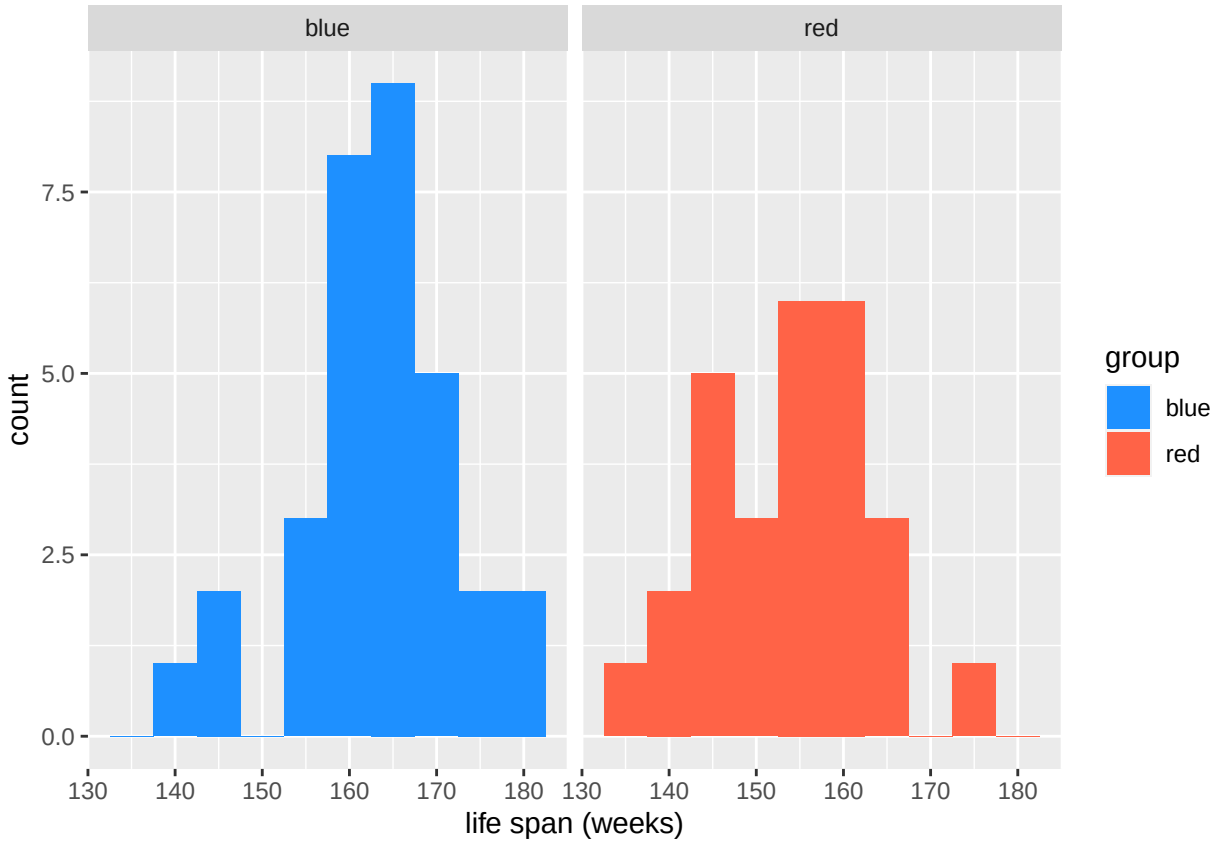


if data frame, ggplot2

```
g <- ggplot(trial,aes(x = treatment,y = pain, colour = treatment))
g <- g + geom_boxplot()
g <- g + geom_jitter()
g
```




```
hamsters <- read.csv("hamsters.csv")
p <- ggplot(hamsters, aes(x=lifespan,fill=group))
p <- p + facet_grid(cols=vars(group))
p <- p + geom_histogram(binwidth = 5)
p <- p + xlab("life span (weeks)")
p <- p + scale_fill_manual(values=c("dodgerblue", "tomato1"))
p
```



more on ggplot2

```
# multiple curves
```

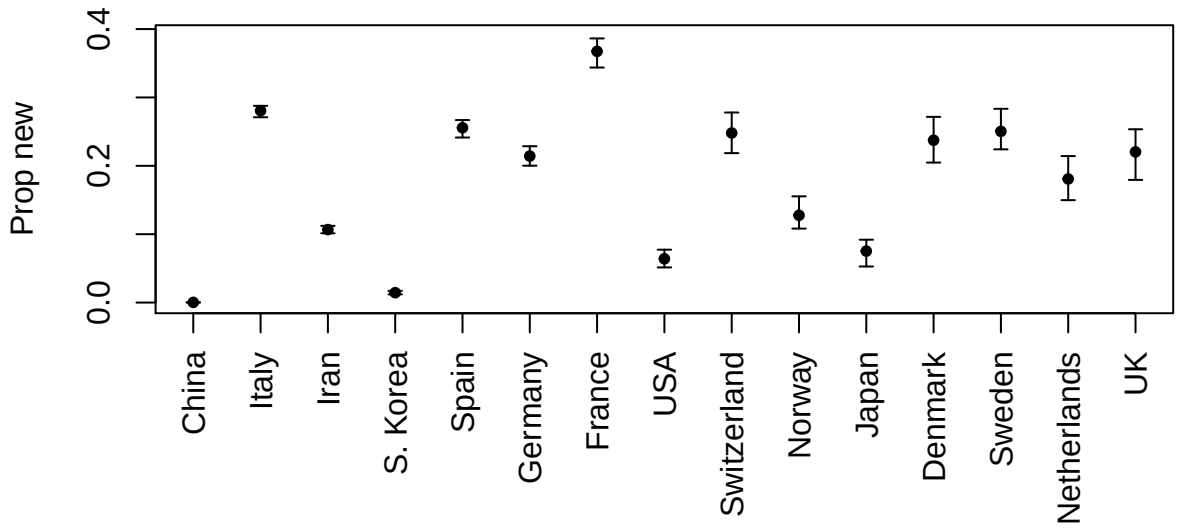
Confidence interval plot

```
covid <- read.csv("coronavirus.csv")
```

```
obs_values<-vector()
lower_cis<-vector()
upper_cis<-vector()
for (a in 1:nrow(covid)){
  country<-covid[a,1]
  obs_total<-covid[a,2]
  obs_new<-covid[a,3]
  obs<-obs_new/obs_total
  old = obs_total - obs_new
  obs_sample<-c(rep("new", obs_new), rep("old", old))
  bootstrap_new<-vector()
  for (b in 1:100){
    bootstrap_sample<-sample(obs_sample, length(obs_sample), replace = T)
    bootstrap_new<-c(bootstrap_new, length(subset(bootstrap_sample, bootstrap_sample == "new")))
  }
  lower_ci<-quantile(bootstrap_new, 0.025)
  upper_ci<-quantile(bootstrap_new, 0.975)
  obs_new<-obs_new/obs_total
```

```
  lower_ci<-lower_ci/obs_total
  upper_ci<-upper_ci/obs_total
  obs_values<-c(obs_values, obs_new)
  lower_cis<-c(lower_cis, lower_ci)
  upper_cis<-c(upper_cis, upper_ci)
}
```

```
ymax<-ceiling(max(upper_cis)*100)/100
par(mar=c(11,4,4,2))
plot(obs_values, xaxt = "n", ylim = c(0,ymax), xlab = "", pch = ".", ylab = "Prop new")
axis(side = 1, at = seq(1,nrow(covid),1), labels = covid$Country, las = 2)
for (a in 1:length(lower_cis)){
  lines(x = c(a,a), y = c(lower_cis[a], upper_cis[a]))
  lines(x = c(a-0.1,a+0.1), y = c(lower_cis[a], lower_cis[a]))
  lines(x = c(a-0.1,a+0.1), y = c(upper_cis[a], upper_cis[a]))
}
points(x = seq(1,nrow(covid),1), y = obs_values, pch = 20)
```



Randomisation

```
# forearms$newGender1 <- sample(forearms$Gender,nrow(forearms), replace=FALSE)
```

Concatenate Strings

```
paste0("a","b")
```

```
## [1] "ab"
```

```
c="Leo"
paste0("I am ",c)
```

```
## [1] "I am Leo"
```

Some functions related to normal distribution

```
rnorm(10) # number,mean(0),sd(1)
```

```
##  [1] -0.26767013 -3.26417857 -0.01590474 -1.28237580  2.57880839 -0.48256762
##  [7]  0.40790757 -0.14279013  0.23027685  1.46574027
```

```
pnorm(1.96) # cumulative distribution function
```

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

```
qnorm(0.975) # the inverse of the second
```

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

Some functions related to uniform distribution

```
runif(26,0,100) # number,min,max
```

```
##  [1] 15.858808 82.700428 22.641563 90.757996 51.139930  0.726628 35.229499
##  [8] 45.692756 25.230992 10.477067 78.201328 97.996398 84.649117 63.664794
## [15] 89.117741 62.809955 17.791786  6.824928 59.174370 73.170481 63.834266
## [22] 53.119184 32.122643 94.791871  9.365563 82.314189
```

Some functions related to t distribution

From t value to p value: x=2.093, df=10 - and +: same show the area t>2.093 and t<-2.093 if significant: one-tailed:<0.1, two tailed<0.05

```
dt(1.729,10) #one tailed
```

```
## [1] 0.09232579
```

```r
dt(2.093,10) #two tailed
```

```
## [1] 0.05275807
```

Hypothesis testing

t-test

Assumptions:

1. Independent random sampling (For independent random sampling, we assume that this assumption has been met from looking at the experimental design.)
2. Normal distribution of data in both groups
3. Sample sizes are small(n $<$ 100)

one-tailed vs two-tailed

```r
# one-tailed (alternative=greater,less)
t.test(n,mu=6,alternative="greater")
```

```
##
##  One Sample t-test
##
## data:  n
## t = -5.0113, df = 19, p-value = 1
## alternative hypothesis: true mean is greater than 6
## 95 percent confidence interval:
##  4.644851      Inf
## sample estimates:
## mean of x
##  4.992489
```

```r
# two-tailed (alternative=two.sided)
t.test(n,mu=6,alternative="two.sided")
```

```
##
##  One Sample t-test
##
## data:  n
## t = -5.0113, df = 19, p-value = 7.752e-05
## alternative hypothesis: true mean is not equal to 6
## 95 percent confidence interval:
##  4.571692 5.413286
## sample estimates:
## mean of x
##  4.992489
```

one sample vs two sample(mu)

```r
t.test(n,mu=6)
```

```
##
##  One Sample t-test
##
## data:  n
## t = -5.0113, df = 19, p-value = 7.752e-05
## alternative hypothesis: true mean is not equal to 6
## 95 percent confidence interval:
##  4.571692 5.413286
## sample estimates:
## mean of x
##  4.992489
```

```r
l <- seq(1,20)
t.test(l,n)
```

```
##
##  Welch Two Sample t-test
##
## data:  l and n
## t = 4.116, df = 19.877, p-value = 0.0005422
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  2.715249 8.299773
## sample estimates:
## mean of x mean of y
## 10.500000  4.992489
```

paired vs unpaired(paired)

```r
c <- rep(1,10)
t.test(c,d,paired=TRUE)
```

```
##
##  Paired t-test
##
## data:  c and d
## t = NaN, df = 9, p-value = NA
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  NaN NaN
## sample estimates:
## mean of the differences
##                       0
```

Power of t test

statistical power is the likelihood that a study will detect an effect when there is an effect there to be detected.

sample size/power, delta: difference in means, type: t-test type, alternative:

```r
power.t.test(n=10, delta = 3, sd =10, sig.level = 0.05, type = "one.sample",
             alternative = "one.sided")
```

```
##
##      One-sample t test power calculation
##
##               n = 10
##           delta = 3
##              sd = 10
##       sig.level = 0.05
##           power = 0.2216581
##     alternative = one.sided
```

```r
power.t.test(delta = 26, sd =30, sig.level = 0.05, power=0.8, type = "two.sample",
             alternative = "one.sided")
```

```
##
##      Two-sample t test power calculation
##
##               n = 17.18096
##           delta = 26
##              sd = 30
##       sig.level = 0.05
##           power = 0.8
##     alternative = one.sided
##
## NOTE: n is number in *each* group
```

Wilcoxon test

1-sample wilcoxon test

```r
wilcox.test(a,mu=0)
```

```
##
##  Wilcoxon signed rank exact test
##
## data:  a
## V = 1, p-value = 1
## alternative hypothesis: true location is not equal to 0
```

Wilcoxon signed-rank test

```r
wilcox.test(c,d,paired = TRUE)
```

```
## Warning in wilcox.test.default(c, d, paired = TRUE): cannot compute exact p-
## value with zeroes
```

```
##
##  Wilcoxon signed rank test with continuity correction
```

```
##
## data:  c and d
## V = 0, p-value = NA
## alternative hypothesis: true location shift is not equal to 0
```

Wilcoxon Rank Sum test

```
wilcox.test(a,n)
```

```
##
##  Wilcoxon rank sum exact test
##
## data:  a and n
## W = 20, p-value = 0.09524
## alternative hypothesis: true location shift is not equal to 0
```

ANOVA

Formulate null and alternative hypothesis and decide which type of ANOVA to use

Assumptions:

1. Independent random sampling
2. Normality of residuals (residuals: distances from group mean)
3. Equality of Variances

Check assumptions Independent random sampling: we assume that this assumption has been met from looking at the experimental design.

In order to test the other two assumptions, we need to make an ANOVA model.

```
model <- aov(pain~treatment,data=trial)
```

Normality of residuals

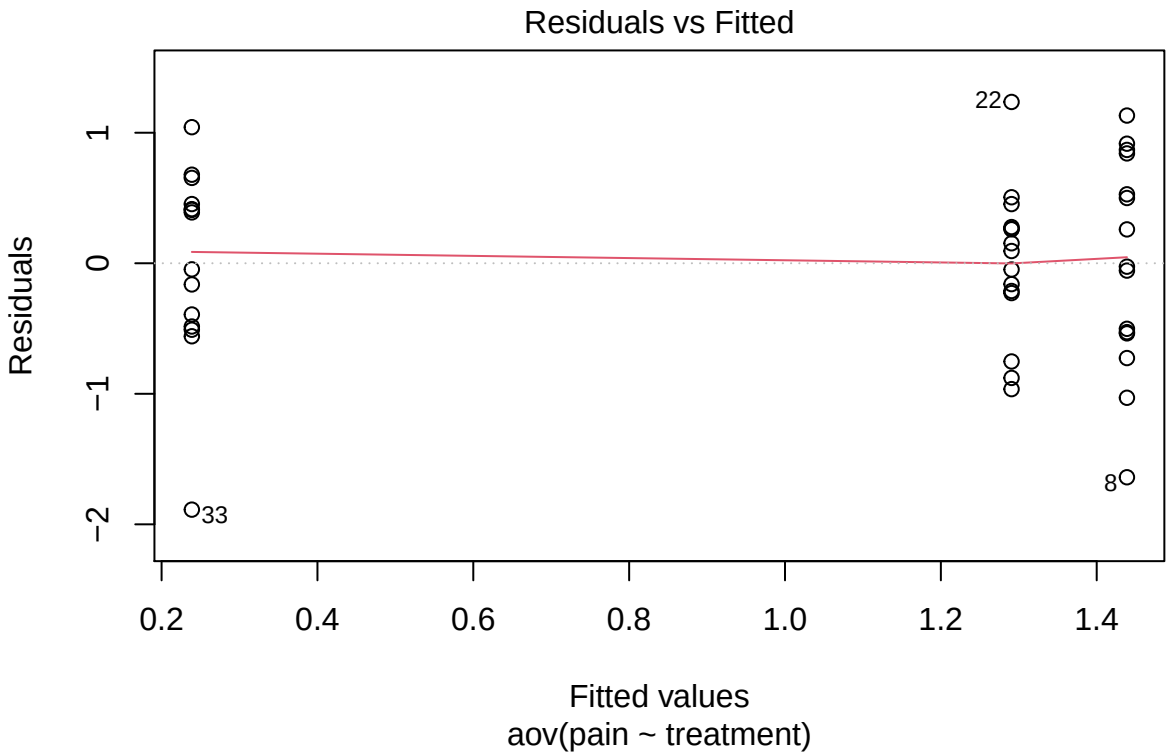
```
shapiro.test(resid(model))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(model)
## W = 0.97317, p-value = 0.3895
```

The null hypothesis of the shapiro-wilk test is that the distribution is normal. The p-value is $>/<$ 0.05, so we can(not) reject the null hypothesis. We conclude that the residuals is (not) normally distributed.

Equality of Variances

```
plot(model,1)
```



If all the "columns" are approximately the same height, the equality of Variances is met. From the "Residual vs Fitted" plot, we conclude that.

Perform ANOVA

We can now perform an ANOVA.

```
summary(model)
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## treatment    2  12.27   6.133    11.9 8.4e-05 ***
## Residuals   41  21.13   0.515
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

conclusion of ANOVA:

Perform TukeyHSD test(a type of post hoc test) We can now perform TukeyHSD test to find out what groups are exactly different.

```
TukeyHSD(model)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
```

```
##
## Fit: aov(formula = pain ~ treatment, data = trial)
##
## $treatment
##                       diff        lwr        upr     p adj
## drugB-drugA   -1.0519098 -1.7005747 -0.4032449 0.0008789
## placebo-drugA  0.1480734 -0.4893095  0.7854563 0.8393996
## placebo-drugB  1.1999832  0.5513183  1.8486481 0.0001610
```

conclusion of TukeyHSD test:

About masking

(from formative)This is a type of masking to avoid any kind of human bias in the experiment or the data analysis. If you have ideas about what you would expect the lifespan of both populations to be, you may be tempted to throw out data points that don't fit in your theory as "outliers", for instance.

Correlation and linear regression

import sample data

```
gbp_malaria <- read.csv("IHME-GBD_2019_DATA_Malaria.tsv", sep = "\t", header = TRUE,
                        stringsAsFactors = FALSE)
gbp_malaria %>%
  group_by(measure_name) %>%
  top_n(2) %>%
  arrange(measure_name)
```

```
## Selecting by val
```

```
## # A tibble: 6 x 4
## # Groups:   measure_name [3]
##   measure_name location_name  year       val
##   <chr>        <chr>         <int>     <dbl>
## 1 Deaths       Nigeria        2008   280604.
## 2 Deaths       Nigeria        2009   276715.
## 3 Incidence    Nigeria        2011 63597364.
## 4 Incidence    Nigeria        2010 64045849.
## 5 Prevalence   Nigeria        2005 50823690.
## 6 Prevalence   Nigeria        2006 51296796.
```

Assumptions (from ADS2 week 18 lecture, slide 18):

1. The residuals are normally distributed.
2. The errors are independent (independent random sampling)
3. The relationships are linear.

We can see there is linear relationship (not "U" shaped etc..) between year and prevalence in all age groups. Also visual inspection shows that there is no obvious outliers. We think it is appropriate to do linear regression.

Although the linear regression lecture mentions the assumption for linear regression is that the residuals need to be normally distributed. However, due to the small sample size, it is quite likely that the residuals are not normally distributed. We think r.squared is enough to assess whether the linear model fits our data points.

r.squared describes the degree of interpretation of input variables to output variables. Formula: In linear regression, the larger the R-square (the closer to 1), the better the linear regression is.

We want to set the threshold of r.squared to 0.7 (from ADS2 week 18 lecture, slide 21). If r.squared is larger than 0.7, we think the result of linear regression can reflect our data points.

Perform linear regression

```
# fit <- lm(val~year,ayus1[[cnt]])
# slope[j,i] <- as.numeric(fit$coefficients[2])
# slope[j,i+1] <- summary(fit)$r.squared
```

Calculate the correlation coefficient

```
# cor()
```

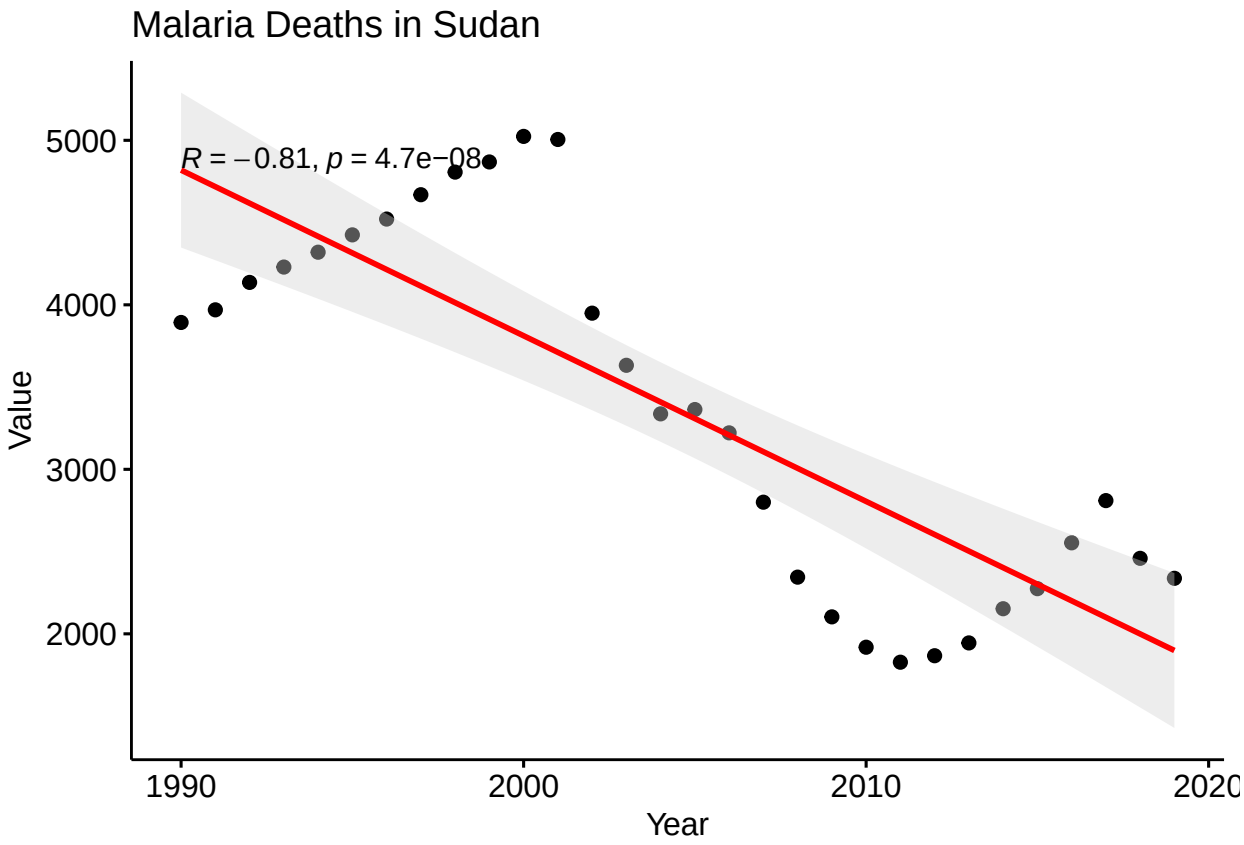
Add straight line to the plot

```
# abline(fit,col="red") fit=lm results
```

Add straight line and confidence interval using ggplot2

```
# Add regression line: add = "reg.line"
# Add confidence interval: conf.int = TRUE
# Add parameter label: stat_cor(method="pearson")
threecountries <- sample(gbp_malaria$location_name, 3, replace = FALSE)
gbp_malaria %>%
  filter(location_name == threecountries[1]) %>%
  filter(measure_name == "Deaths") %>%
  ggscatter(x = "year", y = "val", add = "reg.line",
            add.params = list(color = "red", fill = "lightgray"),conf.int = TRUE) +
  labs(x = "Year", y = "Value", colour = "") +
  stat_cor(method = "pearson") +
  ggtitle(paste0("Malaria Deaths in ", threecountries[1]))
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Bootstrapping

Case resampling (without absolute)

```
# all <- c()
# for (i in 1:10000){
#   ab <- sample(a$points,4,replace=TRUE)
#   ac <- sample(a$points,5,replace=TRUE)
#   all <- c(all,median(ab)-median(ac))
# }
# hist(all)
# exact <- median(Female$points)-median(Male$points)
# mean(all>=exact)
```

Confidence interval

```
# obs_values<-vector()
# lower_cis<-vector()
# upper_cis<-vector()
# for (i in big$percent.obese){
#   # generate sample dataset
#   obs_sample <- c(rep("obese", i*1000/100),
#                   rep("not_obese", (1000-i*1000/100)))
#   bootstrap_new <- c()
```

```
#   # bootstrapping for 100 times
#   for (b in 1:100){
#     bootstrap_sample <- sample(obs_sample, length(obs_sample), replace = T)
#     bootstrap_new <- c(bootstrap_new,
#                        length(subset(bootstrap_sample,
#                                      bootstrap_sample == "obese")))
#   }
#   # get 95% confidence interval
#   lower_ci <- quantile(bootstrap_new, 0.025)
#   upper_ci <- quantile(bootstrap_new, 0.975)
#   lower_cis <- c(lower_cis, lower_ci/1000*100)
#   upper_cis <- c(upper_cis, upper_ci/1000*100)
# }
# big$upper <- upper_cis
# big$lower <- lower_cis
```

Some data cleaning function

gather

```
data <- read.csv("really_tiny_dataset.csv",header = F)
data[c(1,2),c(1,2,784,785)]
```

```
##   V1 V2 V784 V785
## 1  5  0    0    0
## 2  0  0    0    0
```

```
names(data) <- c("num",seq(1,784))
# c(-num): gather according column (which column is x), y is the header
data <- gather(data,key="order",value="intensity",c(-num))
head(data)
```

```
##   num order intensity
## 1   5     1         0
## 2   0     1         0
## 3   4     1         0
## 4   1     1         0
## 5   9     1         0
## 6   2     1         0
```

```
# aline
```

aggregate (aggregate according to column)

```
data_aggr <- aggregate(intensity ~ num+order,data=data,mean)
head(data_aggr)
```

```
##   num order intensity
## 1   0     1         0
## 2   1     1         0
```

```
## 3   2     1         0
## 4   3     1         0
## 5   4     1         0
## 6   5     1         0
```

Categorical data

chi-square

input data

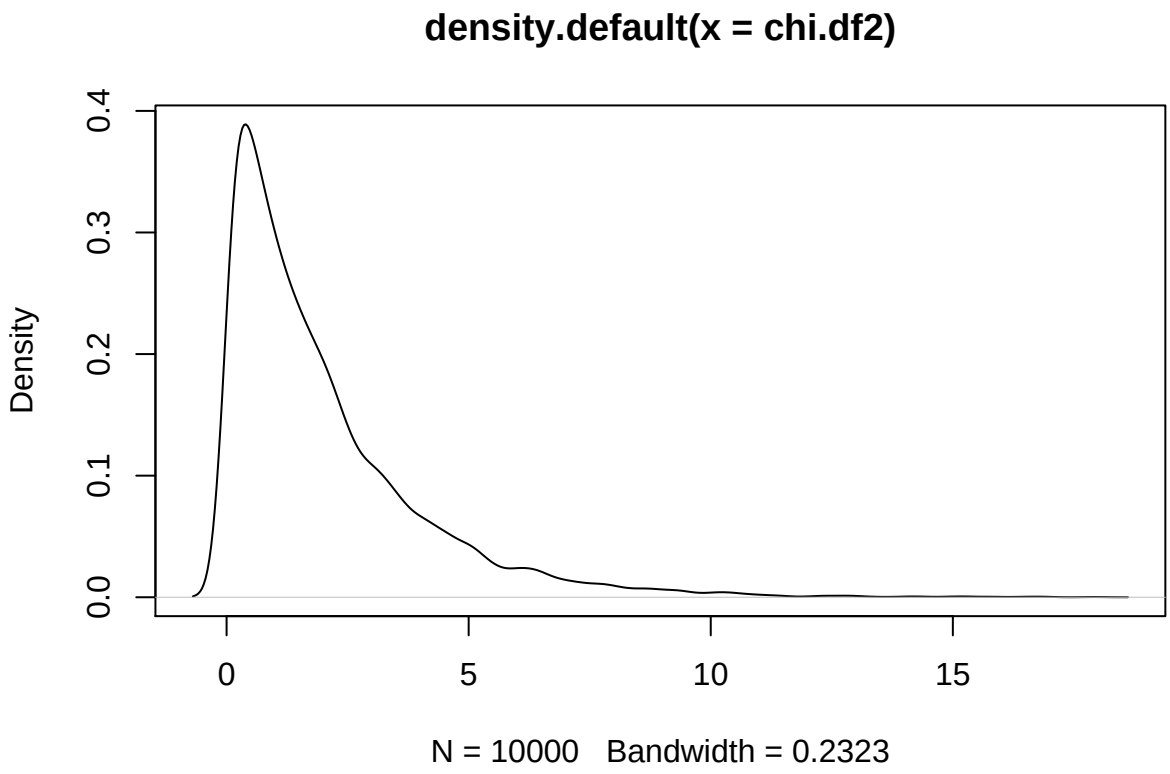
```
survey <- matrix(c(84,82,34,82,57,11),nrow = 3)
row.names(survey) <- c("Good", "Fair", "Bad")
colnames(survey) <- c("A","B")
survey_df <- as.data.frame(as.table(survey))
names(survey_df) <- c("Rating","School","Freq")
```

Assumptions for chi-square test:

1. Discrete, categorical data.
2. No expected cell frequencies are less than 1.
3. No more than 20% are less than 5.

Generate chi square distribution

```
chi.df2 <- rchisq(10000, df=2)
plot(density(chi.df2))
```



Goodness of fit (actual vs expected)

```
# Given probability (expected)
chisq.test(c(84,82,34), p=c(0.45,0.43,0.12))
```

```
##
##  Chi-squared test for given probabilities
##
## data:  c(84, 82, 34)
## X-squared = 4.7527, df = 2, p-value = 0.09289
```

```
# ggplot(data=dat, aes(x=chi2.value, color=df)) + geom_density()
```

homogeneity (2-way, same or different?)

```
chisq.test(survey)
```

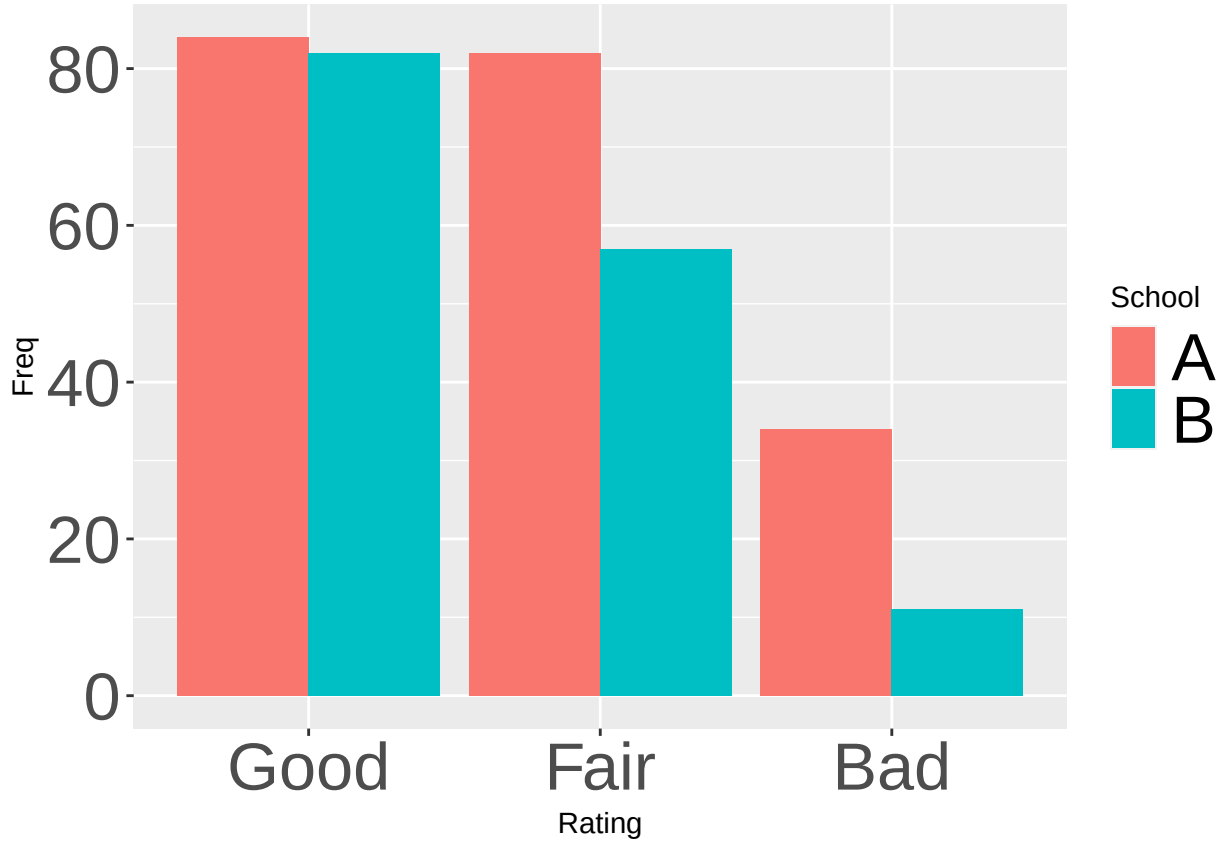
```
##
##  Pearson's Chi-squared test
##
## data:  survey
## X-squared = 9.3235, df = 2, p-value = 0.00945
```

independency (2-way) (similar to homogeneity)

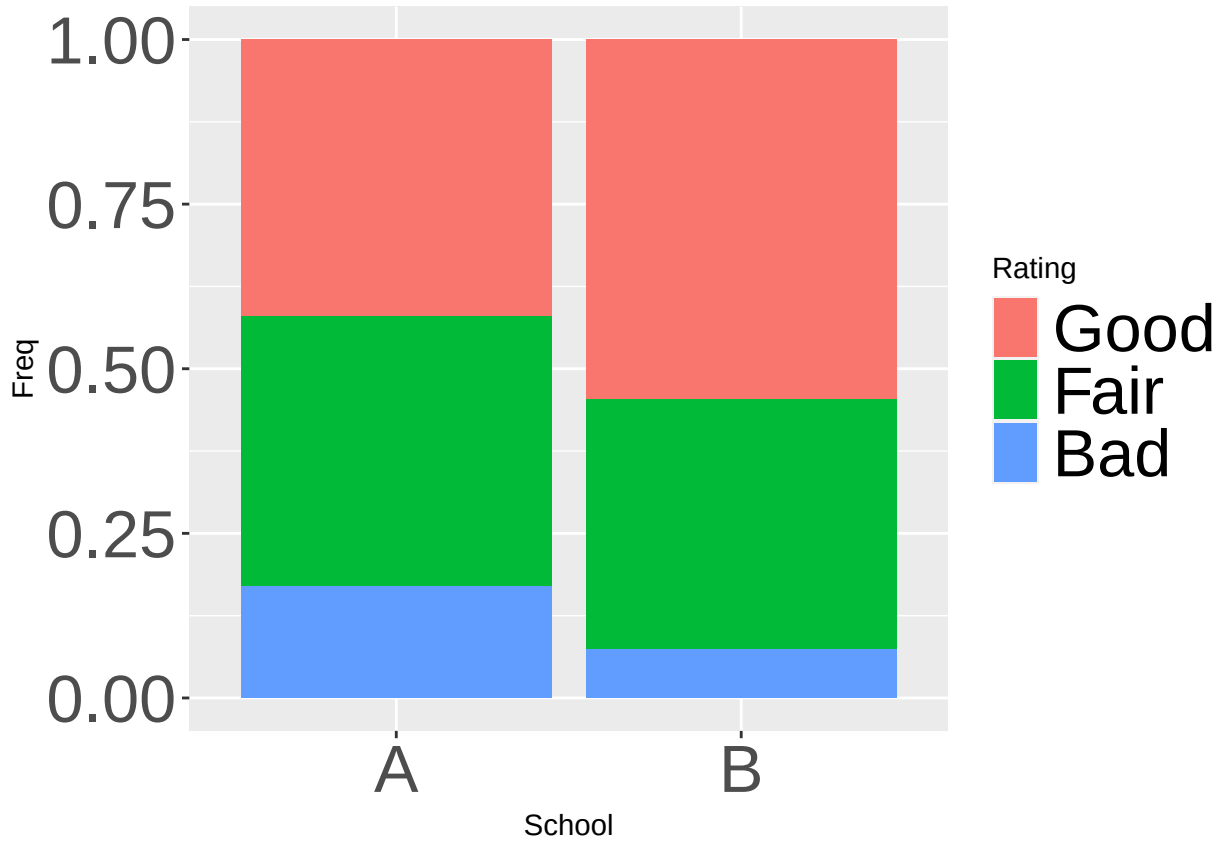
interdependency (3-way, r*cl*)

Data visualization

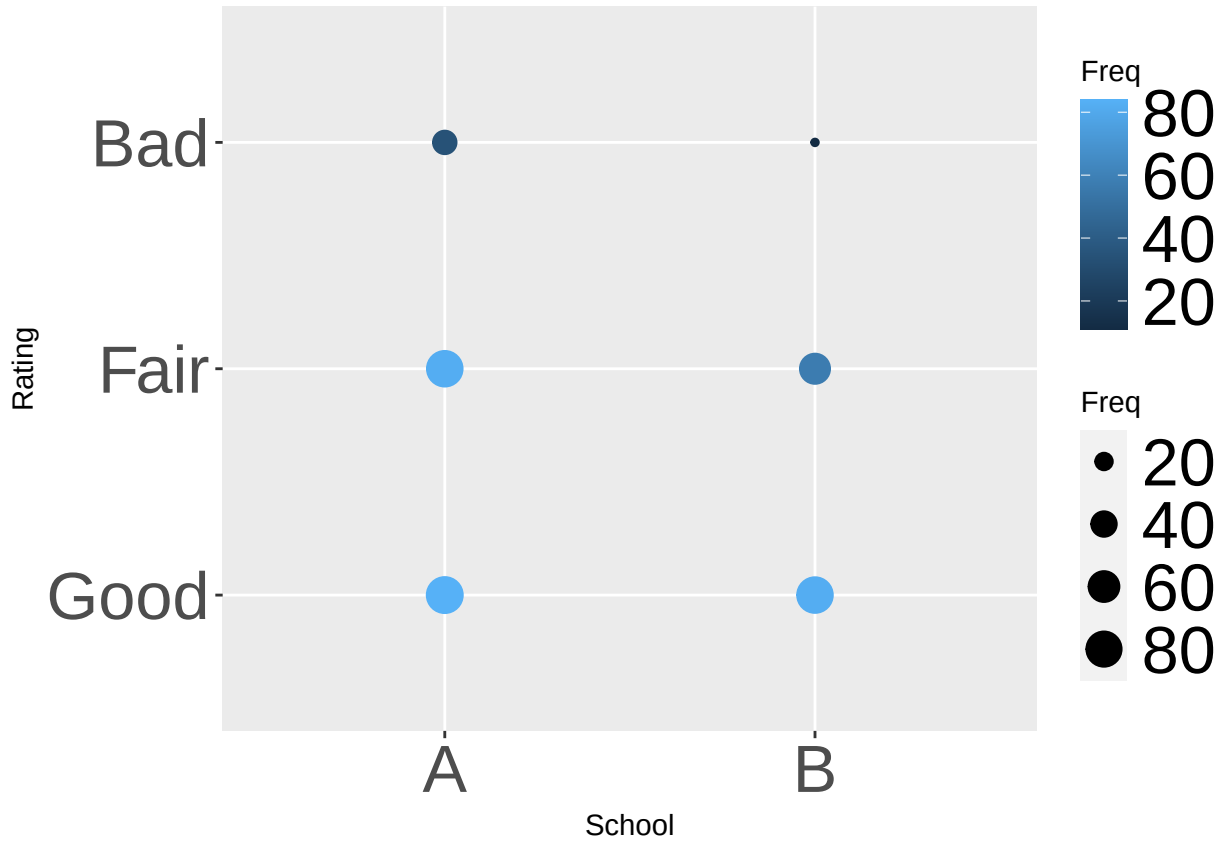
```
#bar plot
ggplot(data=survey_df,aes(x=Rating, y=Freq, fill=School)) +
  geom_bar(stat="identity",position = "dodge") +
  theme(axis.text = element_text(size = 25),
        legend.text =element_text(size = 25))
```



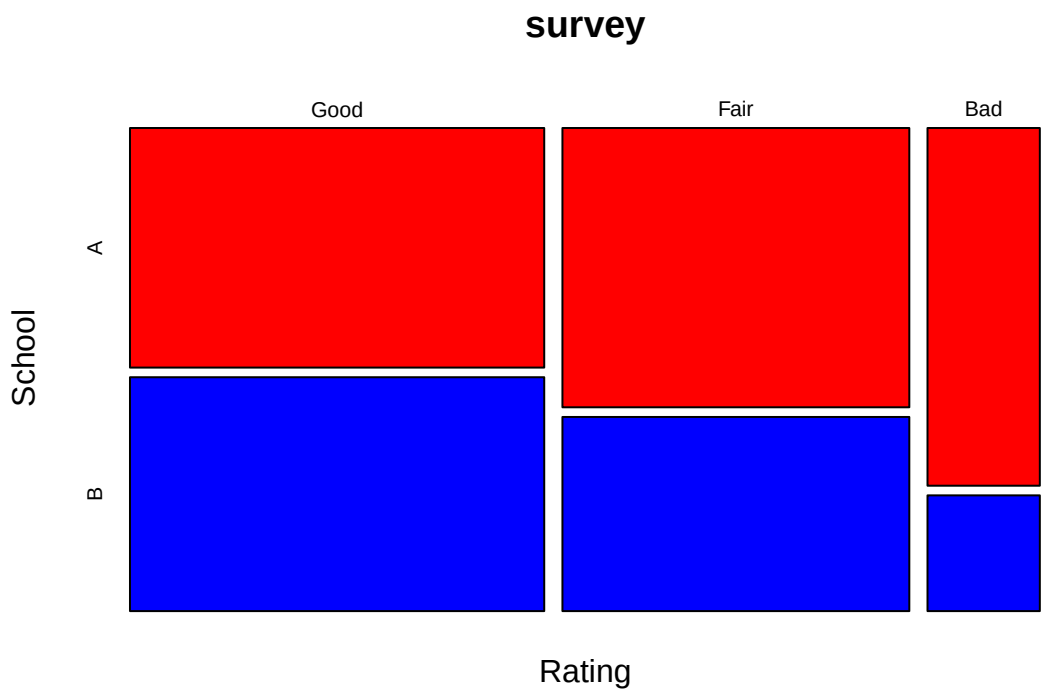

```
#stacked bar plot
ggplot(data=survey_df, aes(x=School, y=Freq, fill=Rating)) +
  geom_bar(stat = "identity",position = "fill") +
  theme(axis.text = element_text(size = 25),
        legend.text =element_text(size = 25))
```




```
#balloon plot
ggplot(data=survey_df, aes(x=School, y=Rating)) +
  geom_point(aes(size=Freq,color=Freq)) +
  theme(axis.text = element_text(size = 25),
        legend.text =element_text(size = 25))
```




```
#mosaicplot
mosaicplot(survey, color = c("red","blue"), xlab="Rating", ylab="School")
```



Fisher's exact test (for small sample size)

```
survival<- matrix(c(7,3,2,7),nrow = 2)
row.names(survival) <- c("Alive","Dead")
colnames(survival) <- c("WT","KO")
#4.2 Chi-square test with Yates's correction on/off
chisq.test(survival, correct = F)
```

```
## Warning in chisq.test(survival, correct = F): Chi-squared approximation may be
## incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  survival
## X-squared = 4.3372, df = 1, p-value = 0.03729
```

```
chisq.test(survival, correct = T)
```

```
## Warning in chisq.test(survival, correct = T): Chi-squared approximation may be
## incorrect
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  survival
## X-squared = 2.6324, df = 1, p-value = 0.1047
```

```
#.4.3 Fisher's exact test
fisher.test(survival)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  survival
## p-value = 0.06978
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##    0.7520079 113.4668907
## sample estimates:
## odds ratio
##   7.166282
```

```
# str_detect & grep
```

group_by (bet mea value of the same group, similar to aggregate)

```
# pixel_summary <- pixels_gathered %>%
#   group_by(x, y, label) %>%
#   summarize(mean_value = mean(value)) %>%
#   ungroup()
```