# C6 Week1 学习笔记

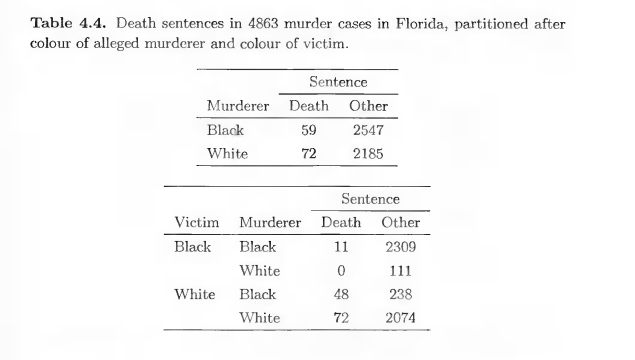
## 1.统计推断introduction

根据样本去推断总体的性质

* 样本是否具有代表性
* 是否有一些没有观察到的隐变量
* 有无因为数据缺失或者实验设计导致的系统性偏差
* 随机化

在进行统计推断时候的的一些常用名词：

* 随机化：当存在一些隐变量(未观察到，但实际上是有影响的)，会造成混杂，这时候必须随机化。比如一个很经典的



第一个表是黑人白人杀人犯，被判刑的案件数，可以考到白人被判死刑的概率大。

第二个表，当把被害人的肤色考虑进去后，分别看会发现：黑人的判死刑的概率明显比白人高。

这里面被害人的肤色其实相当于一个混杂因素。若不注意这个，就会造成完全相反的结果。

* 随机抽样：使得样本更具有代表性。
* 抽样模型：一般比较常用的就是iid，独立同分布
* 假设检验：对一个命题根据样本做统计学意义上的检验
* 置信区间：比如估计学生的平均身高，以95%的置信水平说在某一个区间内[a,b]
* 实验设计：设计实验来最小化偏差
* bootstrap方法
* 置换检验

统计学中两种不同的学派，对应着也有两种不同的统计检验思路：

* 频率学派的频率推断
* 贝叶斯学派的贝叶斯推断

## 2.基本概念

* 概率
* 随机变量
* 密度函数，分布函数/生存函数
* 分位数
* 期望值，方差
* 切比雪夫不等式，6 sigma准则
* 独立事件

R中的manipulate包可以实现一些简单的交互式绘图，添加一些按钮、滚动条啥的。

library(manipulate)

myhist<-function(mu){

hist(iris$Sepal.Length,cl="blue",breaks=30)

lines(c(mu,mu),c(0,10),col="red",lwd=5)

mse=mean((iris$Sepal.Length-mu)^2)

text(6,10,paste("Imbalance = ",round(mse,2)))

}

data(iris)

manipulate(myhist(mu),mu=slider(5,6,step=0.1))

* 条件概率，贝叶斯公式

## 3.diagnostic test

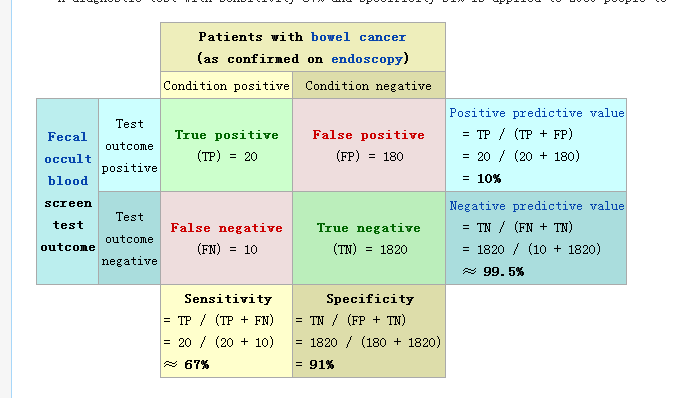
一些基本的概念：这个基本概念很重要

- sensitivity：有病检测出来有病的概率

- specificity：没病检测出来没病的概率

- diagnostic likelihood

一个示例图：



## 4.常见分布、渐进性质

* 大数定律
* consistent 相合性

# week3 学习笔记

## t test

之前写的是按照previus lectures写的，这周开始按照最新的写

data(sleep) head(sleep)

t.test(paired=,var.equal=)

library(datasets)  
library(reshape2)  
data(ChickWeight)  
head(ChickWeight)

## weight Time Chick Diet  
## 1 42 0 1 1  
## 2 51 2 1 1  
## 3 59 4 1 1  
## 4 64 6 1 1  
## 5 76 8 1 1  
## 6 93 10 1 1

#长型数据变为宽型,数据透视表  
wideCW<-dcast(ChickWeight,Chick+Diet~Time,value.var="weight")  
#修改下转换之后的名字  
names(wideCW)[-(1:2)]=paste0("time",names(wideCW)[-(1:2)])  
#  
library(dplyr)

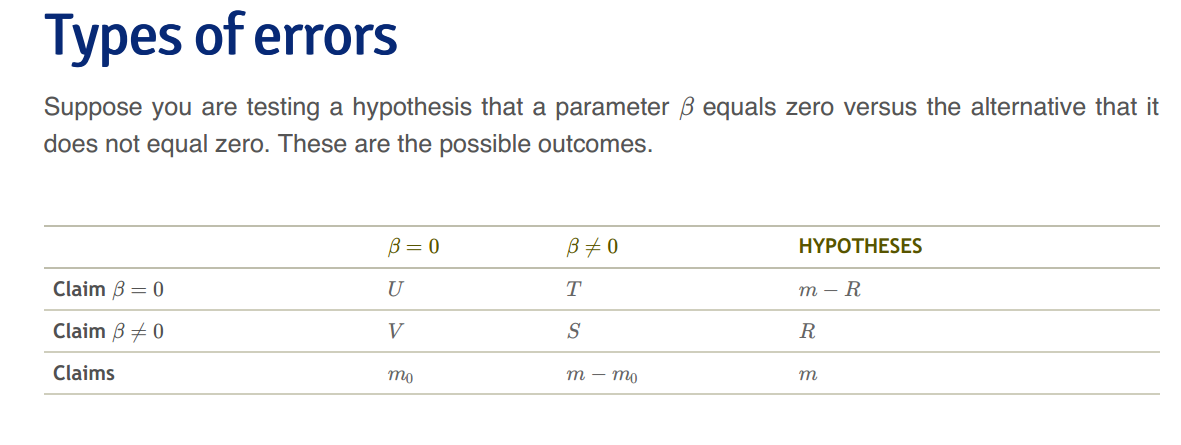
wideCW<-mutate(wideCW,gain=time21-time0)

## hypotheis testing

## p value

# week 4 学习笔记

power.t.test()



* Bonferroni correction:alpha/m 过于保守
* FDR

# 补充资料

## R package——dplyr

##示例数据  
library(hflights)  
##将一些比较大的数据以一个相对简洁的形式展现,  
##妈妈再也不用担心被刷屏了  
hflights\_df<-tbl\_df(hflights)  
head(hflights)  
hflights

常用的几个主要的函数如下：

### 1.filter()

按给定的逻辑判断筛选出符合要求的子数据集, 类似于 base::subset() 函数

filter(hflights\_df, Month == 1)  
##若用R自带的函数写的话就是  
hflights\_df[hflights\_df$Month==1,]

### 2.排列arrange

按照给定的列名依次排序

arrange(hflights\_df, DayofMonth, Month, Year)

### 3.选择: select()

用列名作参数来选择子数据集:

select(hflights\_df, Year, Month, DayOfWeek)

还可以用 : 来连接列名, 没错, 就是把列名当作数字一样使用:

select(hflights\_df, Year:DayOfWeek)

用 - 来排除列名:

select(hflights\_df, -(Year:DayOfWeek))

同样类似于R自带的 subset() 函数

### 4.变形: mutate()

对已有列进行数据运算并添加为新列:

mutate(hflights\_df, gain = ArrDelay - DepDelay, speed = Distance / AirTime \* 60) 作用与 plyr::mutate() 相同, 与 base::transform() 相似, 优势在于可以在同一语句中对刚增加的列进行操作: 而同样操作用R自带函数 transform() 的话就会报错:

### 5.汇总: summarise()

对数据框调用其它函数进行汇总操作, 返回一维的结果:

summarise(hflights\_df, delay = mean(DepDelay, na.rm = TRUE))

等同于 plyr::summarise(), 原文说该函数功能尚不是非常有用, 大概以后的更新会加强吧.

### 分组动作 group\_by()

以上5个动词函数已经很方便了, 但是当它们跟分组操作这个概念结合起来时, 那才叫真正的强大!

当对数据集通过 group\_by() 添加了分组信息后,mutate(), arrange() 和 summarise() 函数会自动对这些 tbl 类数据执行分组操作 (R语言泛型函数的优势).

例如: 对飞机航班数据按飞机编号 (TailNum) 进行分组, 计算该飞机航班的次数 (count = n()), 平均飞行距离 (dist = mean(Distance, na.rm = TRUE)) 和 延时 (delay = mean(ArrDelay, na.rm = TRUE))

planes <- group\_by(hflights\_df, TailNum)  
delay <- summarise(planes,   
 count = n(),   
 dist = mean(Distance, na.rm = TRUE),   
 delay = mean(ArrDelay, na.rm = TRUE))  
delay <- filter(delay, count > 20, dist < 2000)

### 连接符 %.%

包里还新引进了一个操作符, 使用时把数据名作为开头, 然后依次对此数据进行多步操作. 比如: Batting %.% group\_by(playerID) %.% summarise(total = sum(G)) %.% arrange(desc(total)) %.% head(5) 这样可以按进行数据处理时的思路写代码, 一步步深入, 既易写又易读, 接近于从左到右的自然语言顺序, 对比一下用R自带函数实现的: head(arrange(summarise(group\_by(Batting, playerID), total = sum(G)) , desc(total)), 5) 或者像这篇文章所用的方法: totals <- aggregate(. ~ playerID, data=Batting[,c("playerID","R")], sum) ranks <- sort.list(-total