R lecture notes 2023-11-29

Adam Tallman

2023-11-29

knitr::opts\_chunk$set(echo = TRUE)  
library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.3 ✔ readr 2.1.4  
## ✔ forcats 1.0.0 ✔ stringr 1.5.0  
## ✔ ggplot2 3.4.4 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.3 ✔ tidyr 1.3.0  
## ✔ purrr 1.0.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(lattice)  
library(Rling)  
library(languageR)  
library(nhstplot)  
library(reshape)

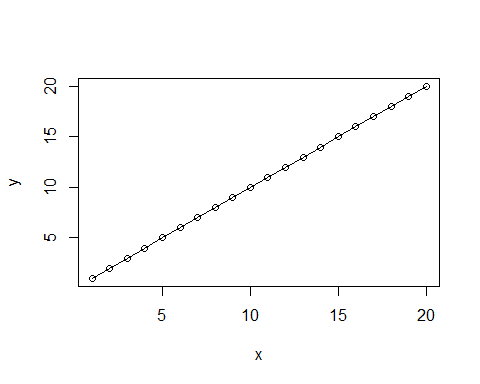
##   
## Attaching package: 'reshape'  
##   
## The following object is masked from 'package:lubridate':  
##   
## stamp  
##   
## The following object is masked from 'package:dplyr':  
##   
## rename  
##   
## The following objects are masked from 'package:tidyr':  
##   
## expand, smiths

elp.df <- read.csv("/Users/Adan Tallman/Desktop/ELP\_full\_length\_frequency.csv")  
senses <- read.csv("/Users/Adan Tallman/Desktop/winter\_2016\_senses\_valence.csv")  
data(ldt)  
data(sharedref)

## Linear models

A deductive versus a statistical model can be seen by simulation.

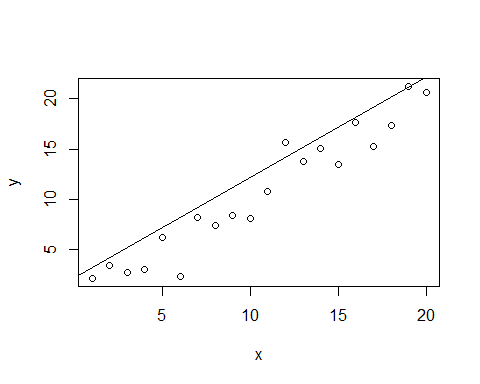
##Deductive model  
x <- seq(from=1, to=20)  
b <- 1  
a <- 0  
y <- a + b\*x  
plot(y~x)+lines(y,x)



## integer(0)

Simulating a stochastic model would look like this.

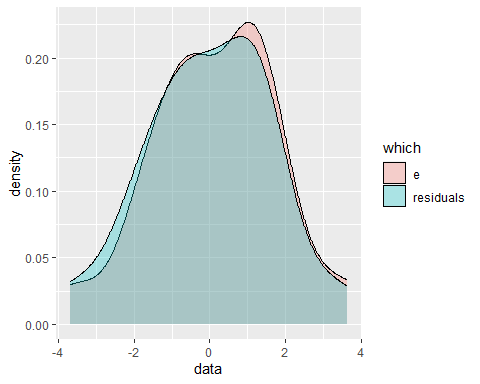
##Statistical model  
set.seed(12345)  
x <- seq(from=1, to=20)  
b <- 1  
a <- 0  
e <- rnorm(m=0,sd=2, n=20)  
y <- a + b\*x + e  
plot(y~x)+abline(y,x)



## integer(0)

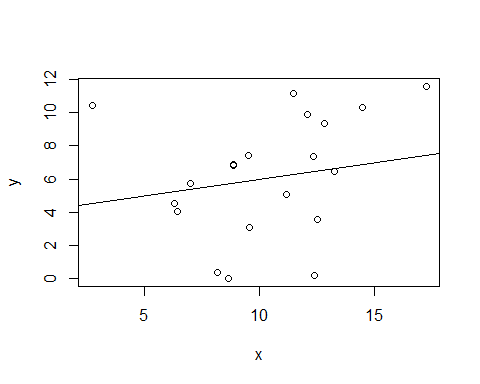
Those error terms are the `residuals" that we discussed last lecture.

y\_hat <- predict(lm(y~x))  
residuals <- y - y\_hat  
  
error\_residuals <- make.groups(e, residuals)  
  
ggplot(error\_residuals, aes(x=data, fill=which))+  
 geom\_density(alpha=0.3)



How do I know that my line is making predictions

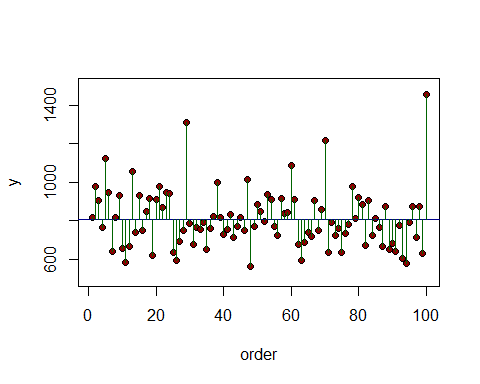
##Line through unrelated vectors  
set.seed(12345)  
x <- rnorm(20, 10, 4)  
y <- rnorm(20, 5, 3)  
plot(y~x)+abline(a=4, b=0.2)



## integer(0)

How do we know when our line explains anything? Let’s look at the ldt database. We can visualize the variance by using a straight line through the datapoints. The x-axis is just the order of elements in the vector.

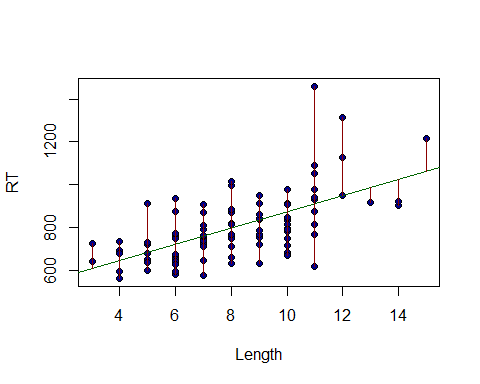
data(ldt)  
RT <- ldt$Mean\_RT  
plot(1:100,RT,ylim=c(500,1500),ylab="y",xlab="order",pch=21,bg="darkred")  
abline(h=mean(RT),col="darkblue")  
for(i in 1:100) lines(c(i,i),c(mean(RT),RT[i]),col="darkgreen")

 You are calculating the overall distance the points are from the mean.

var(RT)

## [1] 23472.17

Length <- ldt$Length  
plot(Length,RT,pch=21,bg="darkblue")  
abline(lm(RT~Length),col="darkgreen")  
fitted <- predict(lm(RT~Length))  
lines(c(0,0),c(12,11.755556))  
for (i in 1:100)  
 lines (c(Length[i],Length[i]),c(RT[i],fitted[i]),col="darkred")



sqrt(sum((RT-fitted)^2))

## [1] 1202.32

I will give an overview of how to read an lm() model below.

model\_1 <- lm(Mean\_RT~Length, data=ldt)  
summary(model\_1)

##   
## Call:  
## lm(formula = Mean\_RT ~ Length, data = ldt)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -291.74 -77.81 -3.69 47.92 546.22   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 498.443 41.949 11.882 < 2e-16 \*\*\*  
## Length 37.644 4.879 7.716 1.02e-11 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 121.5 on 98 degrees of freedom  
## Multiple R-squared: 0.3779, Adjusted R-squared: 0.3716   
## F-statistic: 59.53 on 1 and 98 DF, p-value: 1.019e-11

## Analysis of Variance

head(senses)

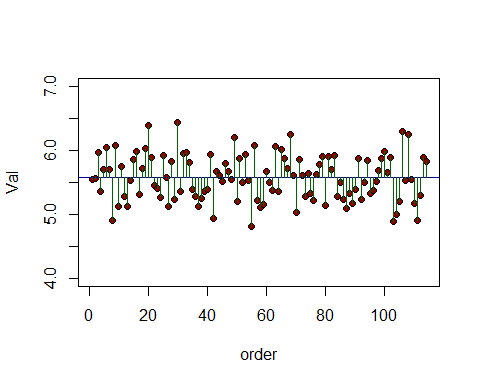
## Word Modality Val  
## 1 abrasive Touch 5.398113  
## 2 absorbent Sight 5.876667  
## 3 aching Touch 5.233370  
## 4 acidic Taste 5.539592  
## 5 acrid Smell 5.173947  
## 6 adhesive Touch 5.240000

tail(senses)

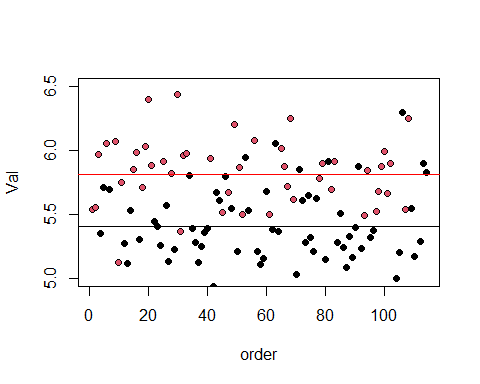
## Word Modality Val  
## 400 wide Sight 5.450416  
## 401 wiry Sight 5.665333  
## 402 wispy Sight 5.678721  
## 403 wizened Sight 5.733333  
## 404 woolly Touch 5.503770  
## 405 yellow Sight 5.451604

How to imagine your null model? It would just be a model that tried to predict every VAL from the mean of all the values. You can visual this as a line going through the mean.

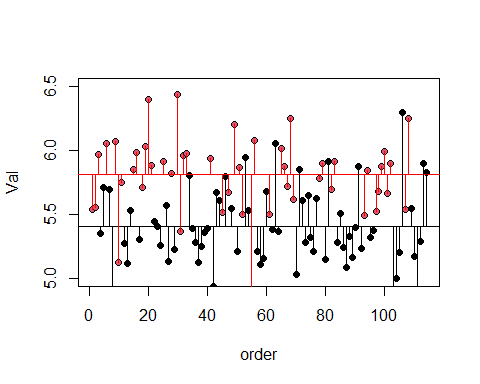
senses\_01 <- filter(senses, Modality =="Taste" | Modality =="Sound")  
modality <- senses\_01$Modality  
Val <- senses\_01$Val  
plot(1:114,Val,ylim=c(4,7),ylab="Val",xlab="order",pch=21,bg="darkred")  
abline(h=mean(Val),col="darkblue")   
for(i in 1:114)   
 lines(c(i,i),c(mean(Val),Val[i]),col="darkgreen")

 The variances are distances from the mean visualized as lines in the plot above. The overall varianxes will be difference if you split the data into groups, which can be visualized as different lines.

senses\_01 <- filter(senses, Modality =="Taste" | Modality =="Sound")  
modality <- senses\_01$Modality  
Val <- senses\_01$Val  
plot(1:114,Val,ylim=c(5,6.5),ylab="Val",xlab="order", pch=21,bg= as.numeric(as.factor(modality)))  
#abline(h=mean(Val[modality=="Sight"]),col="darkgreen")  
#abline(h=mean(Val[modality=="Smell"]), col="darkred")  
abline(h=mean(Val[modality=="Sound"]), col="black")  
#abline(h=mean(Val[modality=="Taste"]), col="cornflowerblue")  
abline(h=mean(Val[modality=="Taste"]), col="red")



senses\_01 <- filter(senses, Modality =="Taste" | Modality =="Sound")  
modality <- senses\_01$Modality  
Val <- senses\_01$Val  
plot(1:114,Val,ylim=c(5,6.5),ylab="Val",xlab="order", pch=21,bg= as.numeric(as.factor(modality)))  
#abline(h=mean(Val[modality=="Sight"]),col="darkgreen")  
#abline(h=mean(Val[modality=="Smell"]), col="darkred")  
abline(h=mean(Val[modality=="Sound"]), col="black")  
#abline(h=mean(Val[modality=="Taste"]), col="cornflowerblue")  
abline(h=mean(Val[modality=="Taste"]), col="red")  
index <- 1:length(Val)  
for (i in 1:length(index)){  
if (modality[i] == "Sound")  
 lines(c(index[i],index[i]),c(mean(Val[modality=="Sound"]),Val  
[i]))  
else  
 lines(c(index[i],index[i]),c(mean(Val[modality=="Taste"]),Val  
[i]), col="red")  
}



The difference can be formalized as the error sum of squares.

Here’s a way of calculating it using base R functions

sound <- senses\_01[senses\_01$Modality=="Sound",]  
taste <- senses\_01[senses\_01$Modality=="Taste",]  
residuals\_Sound <- sound$Val - mean(sound$Val)  
residuals\_Taste <- taste$Val - mean(taste$Val)  
error\_sum\_of\_squares <- sum(residuals\_Sound^2) + sum(residuals\_Taste^2)  
error\_sum\_of\_squares

## [1] 10.13909

The analysis' part of theAnalysis of Variance’ involves comparing this number to the total sum of squares.

total\_sum\_of\_squares <- sum((senses\_01$Val - mean(senses\_01$Val))^2)

The part of the variance that is explained by the different is called the ``treatment sum of squares’, and that’s just the the total sum of squares minus the error sum of squares.

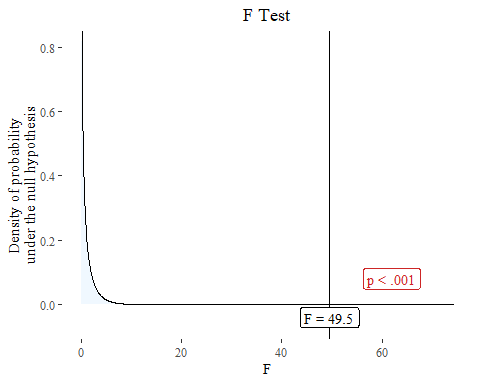
treatment\_sum\_of\_squares <- total\_sum\_of\_squares - error\_sum\_of\_squares

Basically the F-statistic is as follows

You calculate the variance by dividing the sum of squares by their degrees of freedom.

F\_ratio <- treatment\_sum\_of\_squares / (error\_sum\_of\_squares/112)

plotftest(f = 49.54, dfnum = 1, dfdenom = 112)



And there is a function in R that can do this.

summary(aov(Val~Modality, data=senses\_01))

## Df Sum Sq Mean Sq F value Pr(>F)   
## Modality 1 4.485 4.485 49.54 1.65e-10 \*\*\*  
## Residuals 112 10.139 0.091   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## Chi-squared test

Let’s take a look at some of the data from Matthew Dryer’s paper on word order correlations.

adpos <- matrix(c(107,12,7,70),ncol=2,byrow=TRUE)  
rownames(adpos)<-c("PostP","Prep")  
colnames(adpos)<-c("OV","VO")  
adpos

## OV VO  
## PostP 107 12  
## Prep 7 70

wordorder <- cbind(c(107, 7), c(12, 70))  
rownames(wordorder) <- c("Postp", "Prep")  
colnames(wordorder) <- c("OV", "VO")  
wordorder <- rbind(wordorder, c(114,82))  
wordorder <- cbind(wordorder, c(119,77,196))  
rownames(wordorder) <- c("PostP", "Prep", "Column Total")  
colnames(wordorder) <- c("OV", "VO", "Row total")  
wordorder

## OV VO Row total  
## PostP 107 12 119  
## Prep 7 70 77  
## Column Total 114 82 196

Here’s how we create the expected frequencies

E <- cbind(c((114\*119)/196, (114\*77)/196),   
 c(82\*119/196,(82\*77)/196))  
rownames(E) <- c("Postp", "Prep")  
colnames(E) <- c("OV", "VO")  
E

## OV VO  
## Postp 69.21429 49.78571  
## Prep 44.78571 32.21429

E.df <- melt(E)

## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by  
## the caller; using TRUE  
  
## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by  
## the caller; using TRUE

colnames(E.df)<-c("Adposition", "Verb.Object", "Expected.Frequency")  
E.df$Observed.Frequency <- c(107,7,12,70)  
E.df

## Adposition Verb.Object Expected.Frequency Observed.Frequency  
## 1 Postp OV 69.21429 107  
## 2 Prep OV 44.78571 7  
## 3 Postp VO 49.78571 12  
## 4 Prep VO 32.21429 70

E.df$oe <- ((E.df$Observed.Frequency - E.df$Expected.Frequency)^2) / E.df$Expected.Frequency  
sum(E.df$oe)

## [1] 125.5068

chisq.test(adpos)

##   
## Pearson's Chi-squared test with Yates' continuity correction  
##   
## data: adpos  
## X-squared = 122.21, df = 1, p-value < 2.2e-16