UNIVERSITÀ DI TRENTO

Bilancia a due piatti: esperimento di taratura

Tommaso Raffaelli

23-12-2024

Indice

Elenco delle figure

Chiave di verifica: 877b1af3fb86aea58f44c21692279e76cc770098

1 Introduzione

La bilancia a due piatti è uno strumento classico utilizzato per misurare masse, confrontando un oggetto di massa incognita con pesi di massa nota. La precisione di tale strumento dipende dalla corretta taratura, che data dalla relazione tra l'angolo di equilibrio dell'ago (δ) e la differenza di massa (ΔF) tra i due piatti.

Questo esperimento virtuale ha l'obiettivo di raccogliere dati attraverso una serie di misurazioni simulate, analizzarli mediante tecniche di regressione e determinare i parametri caratteristici della bilancia.

Possiamo dividere l'intero processo dell'esperimento in queste fasi:

1. Pianificazione: Definire una serie di combinazioni di $F1$ e ΔF per coprire l'intero intervallo operativo della bilancia.
2. Acquisizione Dati: Per ogni combinazione di $F1$ e ΔF, registrare l'angolo di equilibrio fornito dalla simulazione.
3. Analisi dei Dati: Applicare una regressione sui dati trasformati, utilizzando la modello fisico e determinare i parametri a e FM che meglio descrivono i dati raccolti.
4. Valutazione del Modello: Analizzare i residui per valutare la bontà dell'adattamento del modello ai dati.

2 Pianificazione dell'esperimento

Come detto in precendenza durante la taratura possiamo modificare due variabili del sistema **forza sul piatto sinistro** che chiameremo `Forza_sx` e **differenza fra i due piatti** che verrà chiamata `Delta_forza`. Deciso questo possiamo generare una tabella di test che verrà data all'operatore che svolgerà le misurazioni necessarie per la calibrazione

Durante l'aquisizione dei dati per la taratura l'operatore può modificare due variabili del sistema:

- `Forza sul piatto sinisro`
- `Differenza di forza fra i due piatti`

La prima, `Forza_sx`, varia fra 100 e 500 a intervalli di 50, invece la seconda, `Delta_forza`, varia fra -50 e 50 a intervalli di 5.

Sapendo tutto questo possiamo creare la tabella di test usando la funzione `expand.grid`.

```
Ord_dati <- expand.grid(
  Forza_sx = seq(100, 500, 50),
  Delta_forza = seq(-50, 50, 5),
  Risultato = NA
) %>% mutate(
  StdOrd = 1:n(),
  RunOrd = sample(n()),
  .before = Forza_sx
) %>% arrange(RunOrd)

# write.csv(Ord_dati, "Test_suite.csv")
```

Definiamo `StdOrd` come ordine in cui `expand.grid` ha generato la tabella e `RunOrd` come una **casualizzazione** di `StdOrd` in modo da ottenere una sequenza di misure casualizzate così da diminurire l'effetto di di possibili variabili modificanti che non vendono considerate nel modello fisico.

Ottenendo la seguente tabella: 2.1:

```
Ord_dati[1:10,] %>%
  kable(booktabs=T, caption="Ordine di test") %>%
  kable_styling(latex_options = c("striped", "HOLD_position"), position = "center")
```

Tabella 2.1. Ordine di test

StdOrd	RunOrd	Forza_sx	Delta_forza	Risultato
100	1	100	5	NA
148	2	250	30	NA
70	3	400	-15	NA
51	4	350	-25	NA
65	5	150	-15	NA
136	6	100	25	NA
39	7	200	-30	NA
125	8	450	15	NA
91	9	100	0	NA
77	10	300	-10	NA

Ora l'addetto può procedere con l'aquisizione dei dati.

Come si può notare la funzione `expand.grid` mette all'interno della tabella tutte le combinazioni delle due variabili `Forza_sx` e `Delta_forza`, normalmente in un esperimento, non è sempre necessario svolgere tutte le prove possibili poichè possiamo ovviare alla mancanza di dati con metodi dell'analisi statistica, nel caso di questo esperimento essendo virtuale e non richiedendo un' enorme quantità di tempo sono state prese misure per tutte e 192 le combinazioni all'interno della tabella, in un caso reale questa questione sarebbe da riconsiderare.

3 Lettura dei dati

Come è già stato detto durante la Pianificazione dell'esperimento dall'addetto otterremo un file `csv` che possiamo convertire in una tabella 2.1 contenente:

- `time`
- `F1`
- `DF`
- `angle`

Tabella dei dati ottenuti:

```
Misurazioni <- read.csv("misurazioni.csv", comment="#")

Misurazioni[1:10,] %>%
  kable(booktabs=T, caption="Ordine di test") %>%
  kable_styling(latex_options = c("striped", "HOLD_position"), position = "center")
```

Tabella 3.1. Ordine di test

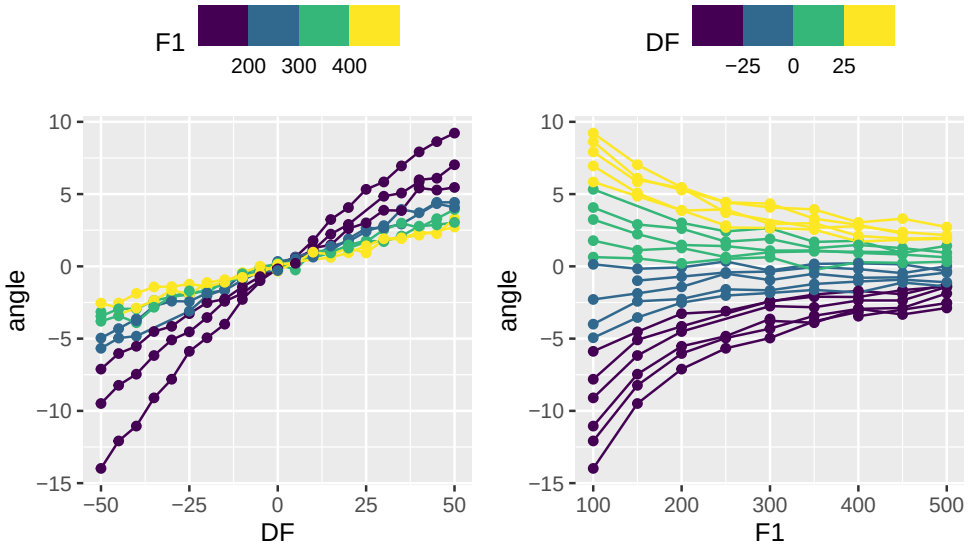
time	F1	DF	angle
0	100	5	0.64
60	250	30	2.82
120	400	-15	-1.05
180	350	-25	-1.95
240	150	-15	-2.42
300	100	25	5.33
360	200	-30	-4.15
420	450	15	0.82
480	100	0	0.14
540	300	-10	-0.95

Mettendo a grafico otteniamo:

```
g1 <- Misurazioni %>%
  ggplot(aes(x=DF, y=angle, group=F1, color=F1)) +
  geom_line() +
  geom_point(aes(y=angle)) +
  scale_color_viridis_b() +
  theme(legend.position="top")

g2 <- Misurazioni %>%
  ggplot(aes(x=F1, y=angle, group=DF, color=DF)) +
  geom_line() +
  geom_point(aes(y=angle)) +
  scale_color_viridis_b() +
  theme(legend.position="top")

g1 + g2
```



4 Regressione

Ora che abbiamo i dati possiamo iniziare a fare la regressione, ma prima bisogna fare alcune considerazioni del tipo:

- verifica di normalità e verifica delle anomalie
- selezione dei dati di train

e solo dopo possiamo iniziare la regressione.

4.1 Modello fisico

La regressione seguirà il modello fisico:

$$tan(angle) = a\frac{DF}{2F1 + FM + DF}$$

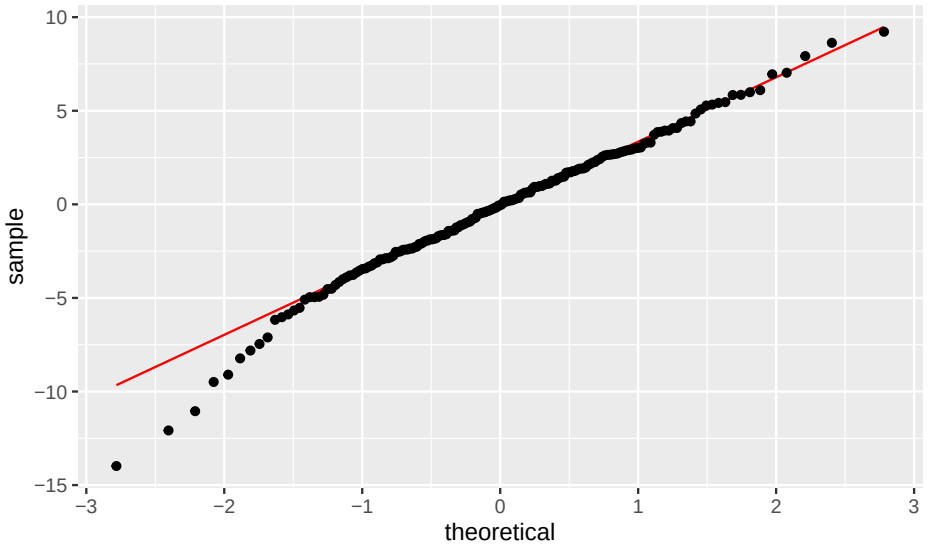
questo non è un modello lineare, infatti, per poter regredire i punti dovremmo usare un metodo non lineare, nel nostro caso useremo il metodo dei minimi quadrati.

4.2 Verifica di normalità

Per poter fare in modo che tutti i test che faremo dopo abbiamo valore dobbiamo prima verificare che i punti, su cui svilupperemo il modello, seguano l'ipotesi di normalità.

La normalità possiamo verificarla in vari modi, in particolare useremo un metodo grafico (`grafico quantile-quantile`) e un test statistico (`test di Shapiro`)

```
Misurazioni %>%
  ggplot(aes(sample=angle)) +
  geom_qq_line(color="red") +
  geom_qq()
```




```r
shapiro.test(Misurazioni$angle)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Misurazioni$angle
## W = 0.98236, p-value = 0.01958
```

Il `test di Shapiro` andrebbe sempre seguito da un test delle anomalie, come `test di Choveniet` o `test di Grubb`, per poter assicurare l'accuratezza del test.

```r
library(outliers)
grubbs.test(Misurazioni$angle)
```

```
##
##  Grubbs test for one outlier
##
## data:  Misurazioni$angle
## G = 3.66348, U = 0.92666, p-value = 0.01783
## alternative hypothesis: lowest value -13.98 is an outlier
```

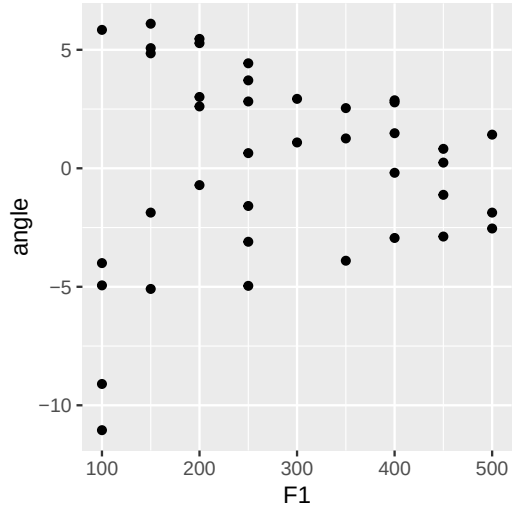
4.3 Selezione dati di train

Prima di fare la regressione possiamo selezionare alcuni punti da omettere dal set di dati usati per la regressione del modello in modo da poterli usare come verifica del modello.

```r
Misurazioni <- Misurazioni %>%
  mutate(
    train = runif(n()) > 1/4
)

g1 <- Misurazioni %>%
  filter(!train) %>%
  ggplot(aes(x=F1, y=angle)) +
  geom_point()

g1 + wrap_table(Misurazioni[1:10,], space="fixed")
```



time	F1	DF	angle	train
0	100	5	0.64	TRUE
60	250	30	2.82	FALSE
120	400	−15	−1.05	TRUE
180	350	−25	−1.95	TRUE
240	150	−15	−2.42	TRUE
300	100	25	5.33	TRUE
360	200	−30	−4.15	TRUE
420	450	15	0.82	FALSE
480	100	0	0.14	TRUE
540	300	−10	−0.95	TRUE

4.4 Regressione ai minimi quadrati

Come già detto la regressione non sarà lineare ma useremo il metodo dei minimi quadrati che in *R* si utilizza tramite la funzione `nls`.

La funzione `nls` deve avere come argomenti:

1. Funzione: Funzione che rappresenta il modello fisico che stimo considerando per la taratura
2. Start: valore di partenza per le variabili di calibrazione `a` e `FM`
3. Data set: misurazioni su cui voliamo regredire il modello fisico

Per comodità possiamo fare una funzione che contiene il Modello fisico e ci restituisce l'angolo, dati i parametri.

```
func <- function(F1, DF, a, FM){
  angle <- (a*(DF / 2*F1 + FM + DF))
  return(angle)
}
```

Ora usando la funzione appena dichiarata possiamo effetturare la regressione del modello.

```
reg <- nls(tan(angle) ~ func(F1, DF, a, FM), start=list(a=1, FM=20), data=Misurazioni)
summary(reg)
```

```
##
## Formula: tan(angle) ~ func(F1, DF, a, FM)
##
## Parameters:
##      Estimate Std. Error t value Pr(>|t|)
## a  -8.588e-05  8.199e-05  -1.047    0.296
## FM -4.745e+03  6.666e+03  -0.712    0.478
##
## Residual standard error: 5.572 on 183 degrees of freedom
##
## Number of iterations to convergence: 2
## Achieved convergence tolerance: 4.08e-08
```

```
cor(atan(Misurazioni$angle), predict(reg)) %>% abs()
```

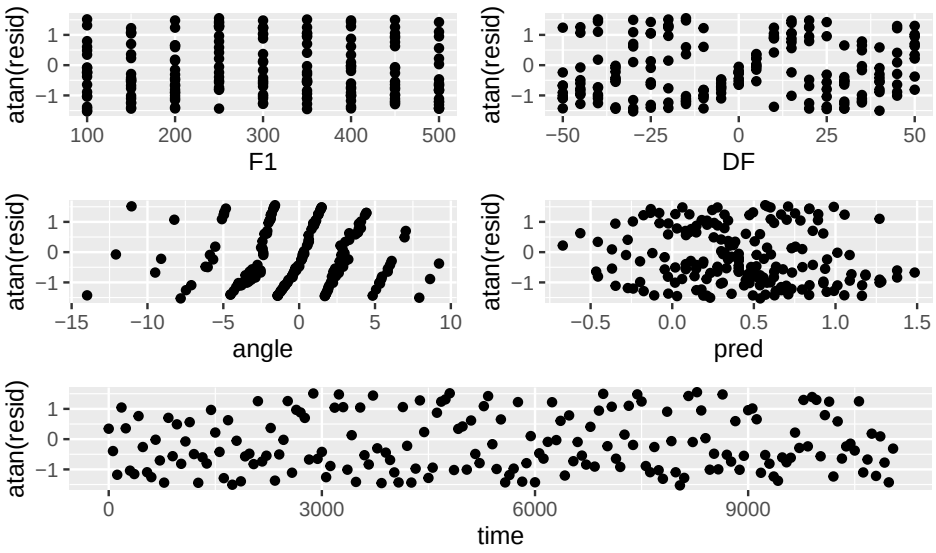
```
## [1] 0.8409558
```

4.5 Analisi dei residui

Un buon metodo per capire se un modello sia adeguato al data set che vogliamo regredire è l'analisi dei residui, essi ci permettono ad esempio di capire, nel caso di un modello lineare, se stiamo facendo underfitting o overfitting.

```
resid_plot <- function(t, n) {
  t %>%
    ggplot(aes(x={{n}}, y=atan(resid))) +
    geom_point()
}


Misurazioni %>%
  add_residuals(reg) %>%
  add_predictions(reg) %>% {
    (resid_plot(., F1) + resid_plot(., DF)) /
    (resid_plot(., angle) + resid_plot(., pred)) +
     resid_plot(., time)
  }
```



Si può notare come nell'grafico dei residui contro l'angolo ci sia un pattern nei punti

4.6 Bootstrap

La tecnica di bootstrap ci permette di ricavare le bande di confidenza del nostro modello

```
stats <- function(data){
  fit <- nls(tan(angle) ~ func(F1, DF, a, FM), data=data, start=list(a=1, FM=20))
  fit$m$getPars()
}

stats(Misurazioni)
```

```
##             a            FM
## -8.587860e-05 -4.744651e+03
```

```
Misurazioni.b <- boot(Misurazioni, R=10000, statistic=function(x, i) stats(x[i,]))
```

```
ci <- list(
  a = boot.ci(Misurazioni.b, type="perc", index=1)$percent[4:5],
  FM = boot.ci(Misurazioni.b, type="perc", index=2)$percent[4:5]
)

ci
```

```
## $a
## [1] -1.907222e-04  1.062504e-05
##
## $FM
## [1] -29463.77  13628.54
```

Ottenendo così gli intervalli di confidenza.

5 Conclusioni

Dall'analisi dei dati raccolti, è emerso che la regressione applicata non è in grado di rappresentare in modo soddisfacente il comportamento reale osservato durante l'esperimento. Tutte le informazioni ricavate, il pattern nei Analisi dei residui, i `p-value` di `a` e `FM` della Regressione ai minimi quadrati e le bande di confidenza ricavate tramite la tecnica di Bootstrap portano a pensare che il problema sia una cattiva regressione.

Le cause di una cattiva regressione possono essere i dati ricavati dall'esperimento o la regressione in sè, questo può significare che non ci sono abbastanza punti, cosa che dubito, o all'interno delle misurazioni ci sono delle anomalie e queste rende la regressione inaccurata, potrebbe essere anche la regressione che non riesce a rappresentare bene tutto quello che succede durante l'acquisizione dei dati