T-test를 이용한 vital linear interpolation의 유효성 검증

**2021. 10. 07. 황치신**

이 글은 **C:\Users\david\GDrive-pub\sql\septic\_shock** 폴더에 위치한   
**011.바이탈에 대한 linear interpolation의 유효성검증.sql** 파일을 Reference로 하여 작성하였다.

사용도구는 **vsCode**와 **PgAdmin4**이다.

본문 및 정리

텍스트이(가) 표시된 사진

자동 생성된 설명

Linear Interpolation은 Vital sign의 각각의 독립변수에 대하여 적용할 만한 보간법이며, 합리적인 해석으로 볼 수 있으나, 반대로 Missing Value들이 평균적인 값들로 추정되어져 Variance가 약화되는 문제점이 있다. Variance는 곧 각각의 Record, Case들의 개성이라 볼 수 있는데 이를 죽여버리는 것이다.

어쨌든 이러한 작업을 통해 만든 Dataset은 과연 유효한지 확인해 볼 필요가 있다.

검증절차는 다음과 같다.

1. missing value가 없는 바이탈 데이터셋(full data)을 준비한다.

2. Full Data로 부터 각각의 바이탈에 대하여 20% 정도의 missing value를 데이터셋을 생성한다.

3. 실제 값과 예측값에 대한 T-test(T검증)을 진행한다.

4. 또한 (실제값-예측값)의 잔차에 대한 분산, 평균, 표준편차를 비교한다.

세부적인 구현은 다음과 같다.

        y0          y         y1

--------t0----------t---------t1---------->t

        |<-- s1   ->|<-  s2 ->|

1. linear interpolation을 위해서는 t-t0 의 간격인 s1, t1-t의 간격인 s2를 설정

2. 데이터셋

    ----------------------------------------------

    -----------input--------|-----output-----------

    t0  t   t1  y0      y1  |      y    predict

    ----------------------------------------------

    1   2   3   100     50  |      71   75

3. 구현 절차

    1. 먼저 t 무작위로 선정

    2. t0 = t-1 ~ t-6 사이의 값으로 무작위 설정(즉, 1시간 ~ 6시간 간격)

    3. t1 = t+1 ~ t+6 사이의 값으로 무작위 설정(즉, 1시간 ~ 6시간 간격)

이건 실제 작성된 코드와 비교하며 진행해보자.

코드

drop table if exists spass\_vitals\_fullfilled\_dataset;

select \*

    into spass\_vitals\_fullfilled\_dataset

from spass\_vitals

where 1=1

and heartrate is not null

and sysbp is not null

and diasbp is not null

and resprate is not null

and tempc is not null

and spo2 is not null

--create index

create index ix\_spass\_vitals\_fullfilled\_dataset\_icustay\_id\_charttime on spass\_vitals\_fullfilled\_dataset(icustay\_id, charttime)

create index ix\_spass\_vitals\_fullfilled\_dataset\_charttime on spass\_vitals\_fullfilled\_dataset(charttime)

* **Spass\_vitals\_fullfilled\_dataset**이라는 테이블을 생성한다. 이 테이블은 **spass\_vitals**라는 basis table로부터 파생된 derived table인데, 여기서 우리가 원하는 6가지 vital 모두 Missing value가 없는 Row들을 검색하고, 이들을 테이블로 생성한다.   
  추가적으로 indexing 작업을 해준다.
* 1. missing value가 없는 바이탈 데이터셋(full data)을 준비한다.

--build dataset

drop table if exists spass\_vitals\_missing\_value\_validation\_linear;

select \*

    ,(select sysbp from spass\_vitals\_fullfilled\_dataset where icustay\_id = d.icustay\_id and charttime = d.t0) as y0

    ,(select sysbp from spass\_vitals\_fullfilled\_dataset where icustay\_id = d.icustay\_id and charttime = d.t) as y

    ,(select sysbp from spass\_vitals\_fullfilled\_dataset where icustay\_id = d.icustay\_id and charttime = d.t1) as y1

    into spass\_vitals\_missing\_value\_validation\_linear

from

(

    select \*

        , (select charttime from spass\_vitals\_fullfilled\_dataset where icustay\_id=c.icustay\_id and charttime >= (c.t - interval '5' hour) and charttime <= (c.t - interval '1' hour) order by random() limit 1) as t0

        , (select charttime from spass\_vitals\_fullfilled\_dataset where icustay\_id=c.icustay\_id and charttime >= (c.t + interval '1' hour) and charttime <= (c.t + interval '5' hour) order by random() limit 1) as t1

    from

    (

        select charttime as t, icustay\_id

        from

        (

            select \*

                ,(select min(charttime) from spass\_vitals\_fullfilled\_dataset where icustay\_id = a.icustay\_id) as first\_charttime

            from spass\_vitals\_fullfilled\_dataset as a

        ) as b

        where 1=1

        and b.charttime > (b.first\_charttime + interval '12' hour)

        order by random()

    ) as c

) as d

where 1=1

and t0 is not null and t1 is not null

order by random()

limit 10000;

* Spass\_vitals\_missing\_value\_validation\_linear라는 테이블을 생성한다. Inline query가 상당하므로 Divide n Conquer 방식으로 가장 기저가 되는 Table부터 접근을 하자.
* select \*
* ,(select min(charttime) from spass\_vitals\_fullfilled\_dataset where icustay\_id = a.icustay\_id) as first\_charttime
* from spass\_vitals\_fullfilled\_dataset as a
* Missing value가 없는 **Spass\_vitals\_fullfilled\_dataset**에서 모든 데이터를 가져오고 추가적으로 **first\_charttime**이라는 column을 생성하였다. **First\_charttime은 각각의 icustay\_id들의 최초 vital이 측정된 시간이다**.
* select charttime as t, icustay\_id
* from
* ( Select … from … ) as b
* where 1=1
* and b.charttime > (b.first\_charttime + interval '12' hour)
* order by random()
* 이 data들 중에서, **vital을 체크한 시간이 최초 시간으로부터 12시간 지난 ROW들만 획득한다**. 이들의 시간을 t 라고 setting하고, 그에 mapping되는 icustay\_id도 보여주며, 이 데이터들을 뒤섞는다. 굳이 icustay\_id를 where에 조건으로 두지 않아도 first\_charttime이 key 역할을 하며 적절한 ROW들을 찾아준다.  
  ICU에 입실한 환자의 vital monitoring setting이 안정화 되어 vital 관측이 원활히 가능하도록 시간을 넉넉히 두고자 하여 12 hour 시간 뒤부터 계측하였다고 판단된다.
* select \*
* , (select charttime from spass\_vitals\_fullfilled\_dataset where icustay\_id=c.icustay\_id and charttime >= (c.t - interval '5' hour) and charttime <= (c.t - interval '1' hour) order by random() limit 1) as t0
* , (select charttime from spass\_vitals\_fullfilled\_dataset where icustay\_id=c.icustay\_id and charttime >= (c.t + interval '1' hour) and charttime <= (c.t + interval '5' hour) order by random() limit 1) as t1
* from
* ( select … from … where … ) as c
* 엄선한 데이터 C 토대로 해당 데이터의 모든 column과 이에 correspond하는 t0 와 t1을 출력한다. T0는 C의 각각의 t 시간으로부터 직전 5~1시간 사이의 시간들 중 무작위로 선정된 임의의 시간이고, t1은 t 시간으로부터 직후 1~5시간 사이의 시간들 중 무작위로 선정된 임의의 시간이다. 무작위는 연구 및 검증에 있어 연구자의 개입을 최소화 시키기 때문에 객관성을 확보할 수 있는 방법이 된다.   
  이렇게 되면 최대 11시간, 최소 3시간의 term이 존재하는 (t0: -5h / t: 0h / t1: +5h) t0, t, t1이 생성된다. 굳이 1~5시간으로 지정한 이유는, 이전의 연구에서 Shock Prediction을 위해 사용한 vital의 Time Series가 (6시간 + onset time) 이었기 때문이다. 이는 타 논문에서도 사용하는, 환자의 vital 변화를 파악하는 적정 시계열로 사용된다.
* drop table if exists spass\_vitals\_missing\_value\_validation\_linear;
* select \*
* ,(select sysbp from spass\_vitals\_fullfilled\_dataset where icustay\_id = d.icustay\_id and charttime = d.t0) as y0
* ,(select sysbp from spass\_vitals\_fullfilled\_dataset where icustay\_id = d.icustay\_id and charttime = d.t) as y
* ,(select sysbp from spass\_vitals\_fullfilled\_dataset where icustay\_id = d.icustay\_id and charttime = d.t1) as y1
* into spass\_vitals\_missing\_value\_validation\_linear
* from
* (select … from … where …) as d
* where 1=1
* and t0 is not null and t1 is not null
* order by random()
* limit 10000;
* t0, t, t1이 존재하는 temp table D로부터, t0, t1이 null 이 아닌 ROW들만 무작위로 10000개 출력한다. 이때 출력하는 데이터는 기존의 column들과, (icustay\_id, charttime)을 candidate key로 하는 y0, y, y1 column이다.  
  위에서는 systolic blood pressure만을 대상으로 검증 테이블을 생성하였다.
* 세부적인 구현은 다음과 같다.
* y0          y         y1
* --------t0----------t---------t1---------->t
* |<-- s1   ->|<-  s2 ->|
* 1. linear interpolation을 위해서는 t-t0 의 간격인 s1, t1-t의 간격인 s2를 설정
* 2. 데이터셋
* ----------------------------------------------
* -----------input--------|-----output-----------
* t0  t   t1  y0      y1  |      y    predict
* ----------------------------------------------
* 1   2   3   100     50  |      71   75
* 3. 구현 절차
* 1. 먼저 t 무작위로 선정
* 2. t0 = t-1 ~ t-6 사이의 값으로 무작위 설정(즉, 1시간 ~ 6시간 간격)
* 3. t1 = t+1 ~ t+6 사이의 값으로 무작위 설정(즉, 1시간 ~ 6시간 간격)
* Spass\_vitals\_missing\_value\_validation\_linear 테이블은 [2]에서 얘기하는 바와 같이 t0, t, t1, y0, y, y1을 갖고 있다.  
  **이제 우리가 알고 싶은 것은 y0, y1으로 linear interpolation을 수행하여, 출력된 predict의 값이 y와 비교하였을 때 유효한 지를 검증하는 것이다**.
* 2. Full Data로 부터 각각의 바이탈에 대하여 20% 정도의 missing value를 데이터셋을 생성한다.

이제 평가 단계로 넘어가자.

--evaluate

select v.\*

    , abs(y-yhat) as residual

from

(

    select u.\*

        , ((t1 - t)/(t1-t0))\*y0 + ((t-t0)/(t1-t0))\*y1 as yhat

    from

    (

        select

             0 as t0

            , (EXTRACT(EPOCH FROM (t - t0))/60.0) as t

            , (EXTRACT(EPOCH FROM (t1 - t0))/60.0) as t1

            , y0, y1, y

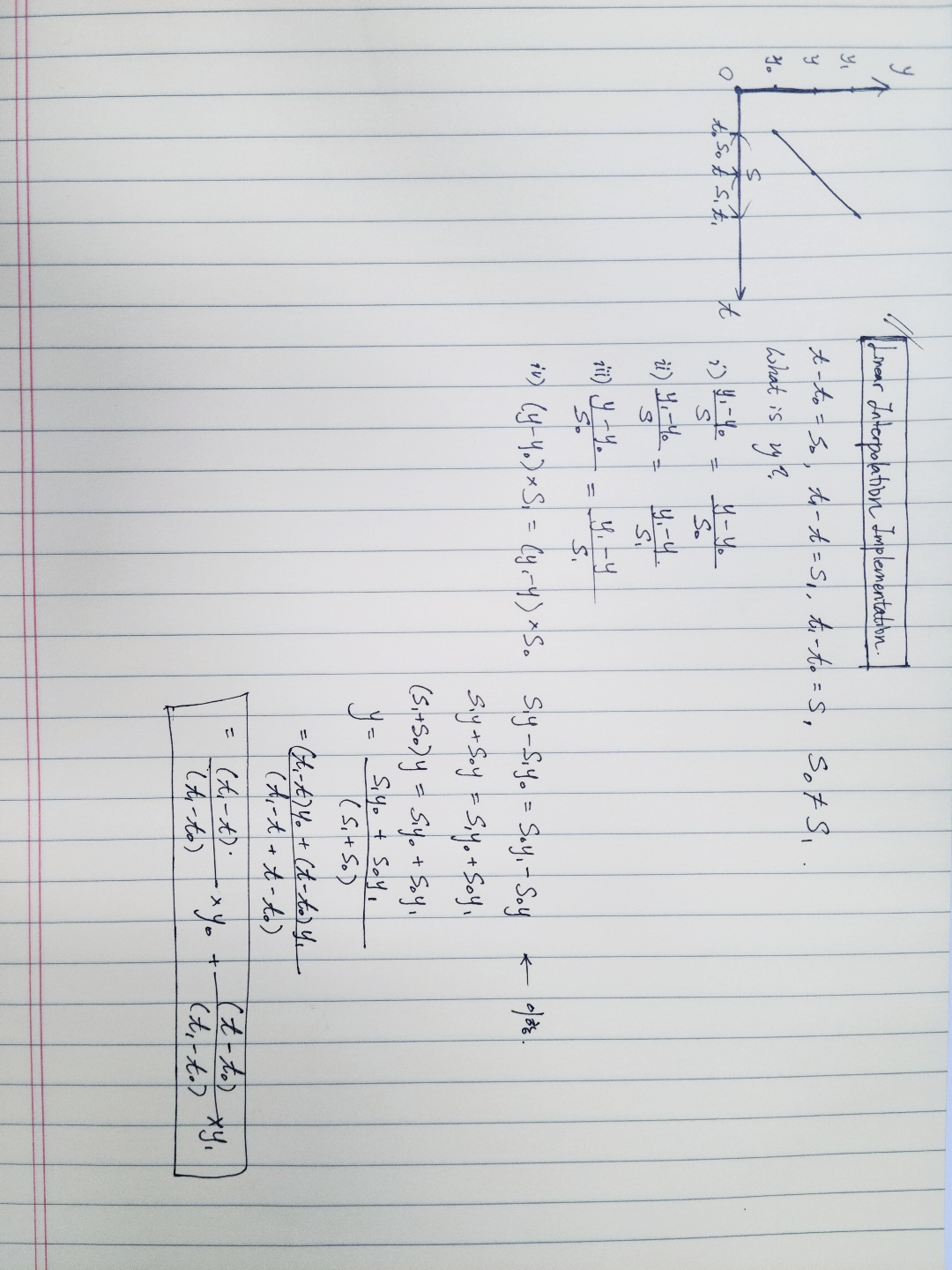
        from spass\_vitals\_missing\_value\_validation\_linear

        limit 10

    ) as u

) as v

* 생성한 테이블로부터 값을 출력한다.  
  t0 column의 모든 값들은 0으로 초기화 시킨다. 이는 이 위에서 무작위로 추출한 ROW들의 관계성을 없애고 각각을 **고유한 검증 데이터**로 사용하기 위해서이다.  
  **(t0, t, t1, y0, y, y1)**  
  t와 t1은 각각, t0 로부터 경과한 minute, t 로부터 경과한 minute을 value로 갖는다.   
  이로 인해 순수하게 **(t0, t, t1, y0, y, y1)** 만을 통한 유효성 검증이 가능하게 처리되었다.
* U 로부터 모든 data를 득하고, linear interpolation을 통한 예측값 yhat을 득한다.



T-test

집합 A: spass\_vitals\_missing\_value\_validation\_linear의 y value

집합 B: spass\_vitals\_missing\_value\_validation\_linear의 y hat value

두 개의 가설 중에 하나를 선택할 수 있게 도와주는 통계적 검정방법이다.

Null Hypothesis(귀무가설, H0)과 Alternative Hypothesis(대립(연구)가설, H1)의 두 가지 가설이 존재한다.

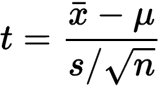
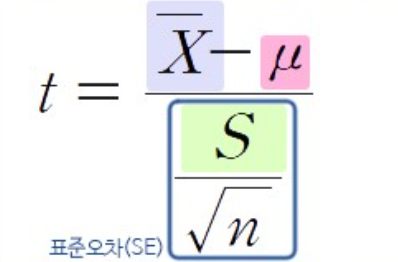
연구자는 두 가지 가설을 설정하게 되며, 하나의 가설 (귀무가설)이 기각되면, 반대 가설(대립가설)이 채택된다는 가설검정 방법이 이것이다.

두 가설 중 하나를 선택하는 것을 돕는 통계적 Tool로써 t 라는 값이 사용되어 t-test라고 한다.

우리는 위 집합 A와 B가 있을 때, 다음과 같은 가설을 하였다.

Null Hypothesis : A와 B는 동일한 집합이다.

Alternative Hypothesis : A 와 B는 다른 집합이다.

  텍스트이(가) 표시된 사진

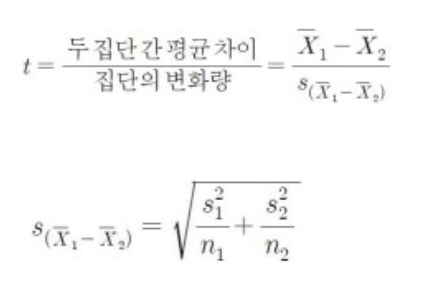
자동 생성된 설명

대부분의 연구의 경우 p-value 0.05를 기준으로 해석하며

만약 p-value가 0.05보다 작으면 귀무가설을 기각하고, 크다면 이를 채택한다.

구하는 법은 <https://www.statisticshowto.com/probability-and-statistics/t-test/> 이거 참고.

<https://m.blog.naver.com/kyoungin90/222081631125> 나는 이걸 참고해서 풀었다.



select v.\*

    , abs(y-yhat) as residual

from

(

    select u.\*

        , ((t1 - t)/(t1-t0))\*y0 + ((t-t0)/(t1-t0))\*y1 as yhat

    from

    (

        select

             0 as t0

            , (EXTRACT(EPOCH FROM (t - t0))/60.0) as t

            , (EXTRACT(EPOCH FROM (t1 - t0))/60.0) as t1

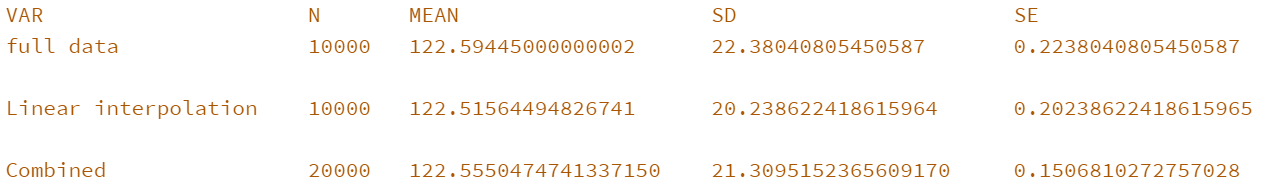
            , y0, y1, y

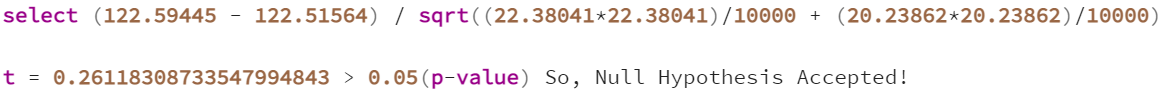
        from spass\_vitals\_missing\_value\_validation\_linear

        limit 10

    ) as u

) as v





그러므로 귀무가설이 유효하다는 것을 통계적으로 검증하였다.

즉, 두 집합 A, B는 동일한 집합이라는 것을 검증한 것이다.

Result

select avg(y), stddev(y), stddev(y)/sqrt(10000)

,avg(yhat), stddev(yhat), stddev(yhat)/sqrt(10000)

from

(

select v.\*

, abs(y-yhat) as residual

from

(

select u.\*

, ((t1 - t)/(t1-t0))\*y0 + ((t-t0)/(t1-t0))\*y1 as yhat

from

(

select

0 as t0

, (EXTRACT(EPOCH FROM (t - t0))/60.0) as t

, (EXTRACT(EPOCH FROM (t1 - t0))/60.0) as t1

, y0, y1, y

from spass\_vitals\_missing\_value\_validation\_linear

limit 20000

) as u

) as v

) as x

HeartRate

Full data 10000 87.83437 17.10604 0.17106

Linear 10000 87.74414 16.02775 0.16028

Combined 20000

t = 0.38491513241459264901

DiasBP

Full data 10000 59.52663 13.53771 0.13538

Linear 10000 59.56785 12.42197 0.12422

t = 0.22434835725063370369

MeanBP

Full data 10000 80.20627 14.40639 0.14406

Linear 10000 80.20715 12.98196 0.12982

t = 0.00453779877137179608 < 0.05

Respirate

Full data 10000 20.51318 6.119914 0.06120

Linear 10000 20.53125 5.577323 0.05577

t = 0.21823436067994337412

Tempc

Full data 10000 37.13097 0.81881 0.008188

Linear 10000 37.12074 0.74880 0.007488

t = 0.921976661060916102335233

Spo2

Full data 10000 97.32790 3.06084 0.030608

Linear 10000 97.34500 2.46345 0.024635

t = 0.43522121077796280414

select abs((97.32790 - 97.34500) / sqrt((3.06084 \* 3.06084)/10000 + (2.46345 \* 2.46345)/10000))