

## Documentation of SPM' s Function

The SPM currently has the following functions:

`Cov_beta`- multiplies a transition function by its associated beta coefficient.

`Wait_tm`- generates a random waiting time.

`Sel_evnt`- selects the event with the smallest waiting time.

`Cos_covs`- multiplies a cost vector by its associated beta coefficient.

`Monthcost`- multiplies month and cost components vector responding to each event that the patients fell into.

`Monthutil`- multiplies month and utility vector responding to each event that the patients fell into.

\*The following examples are based on the patients' characteristics with age as 63, ranking as 1, gender as male, and health state as 4.

### Cov\_beta(x, b, t)

#### **Purpose:**

Replaces the baseline survival function "t" with a patient-specific survival function "newt", where  $newt = t^{\exp(bx)}$

#### **Inputs:**

x- the vector of patient characteristics (#pats x 21), including the column of covariates

When calling the module, the `cos_covs` is a vector of patient characteristics; `age2` is a vector of the covariates. The `covariate_means` is used in for adjusting patients' characteristics. The `covs` replace x.

In the vector of `covariate_mean`, the column order is asy, tia, mi, hs, is, but in the code, we define `oldstate=0, 1,2,3,4` corresponding to asy, tia, mi, hs, and is, respectively. So this is why we use 'oldstate+1' in the `covariate_means`.

```
covs=cos_covs[1]||age2||cos_covs[1,2:20];
covarmn=covariate_means[oldstate+1,];
age2=age**2-covarmn[1]**2;
```

b- the vector of regression coefficients (21 x 6); note: first column of parameter names is not used. They are `tbeta_a`, `tbeta_t`, `tbeta_m`, `tbeta_h`, and `tbeta_i` which are used to modify the transition functions to reflect patient characteristics, of transitions from Asymptomatic, Transient ischemic attack, Ischemic stroke, Myocardial infarction, and Hemorrhagic stroke, respectively.

t- the vector containing the transition function, i.e. baseline survival function (600 x 6); note: first column of the month number is not used. The five baseline survival functions are `tran_a`, `tran_t`, `tran_m`, `tran_h`, and `tran_i` which are the monthly probability of transition from Asy, Tia, Mi, Hs, and Is respectively to each of possible new states.

#### **Input requirement:**

x and b should be the same length.

For t, the intention for the inputs is to be monotonically non-increasing, bounded by 0 and 1.

#### **Returns:**

`newt`- the vector containing patient-specific survival function that is an updated transition function, because we take into account covariates.

---

#### SAS program:

```
start cov_beta(x,b,t);
  exbeta=exp(sum(x#b));
  newt=t##exbeta;
return(newt);
```

```
finish;
```

---

**Example:**

```
Covariate_means:
```

	asy	tia	mi	hs	is
	70.795	69.974	65.220	65.705	70.403
	0.373	0.509	0.661	0.446	0.449
	0.562	0.590	0.682	0.719	0.586
	140.698	150.812	149.052	151.288	156.414
	0.348	0.423	0.273	0.410	0.405
	0.109	0.124	0.173	0.144	0.208
	0.112	0.256	0.369	0.151	0.249
	0.044	0.068	0.037	0.086	0.152
	0.026	0.051	0.079	0.036	0.125
	0.060	0.162	0.133	0.108	0.115
	0.112	0.150	0.102	0.137	0.173
	0.000	0.034	0.000	0.000	1.000
	0.002	0.111	1.000	0.043	0.094
	0.000	0.000	0.000	0.000	0.000
	0.000	1.000	0.000	0.000	0.074
	0.000	0.000	0.000	0.000	0.000
	0.000	0.000	0.000	0.000	0.000
	0.000	0.000	0.000	0.000	0.000
	0.000	0.000	0.000	0.000	0.000
	0.000	0.000	0.000	0.000	0.000
	0.000	0.000	0.000	0.000	0.000

```
covamn=covariate_means[oldstate+1,]=covariate_means[, 5]:
```

```
is
70.403
0.449
0.586
156.414
0.405
0.208
0.249
0.152
0.125
0.115
0.173
1.000
0.094
0.000
0.074
0.000
0.000
0.000
0.000
```

```
cos_covs= age ||pats[i, 2:15]||r1||r2||r3||r4||r5:
```

age	gender	evsmoke	sbp	hrx	diab	cad	af	chf	ic
63	1	0	140	0	0	0	0	0	0
murm	h_tia	h_mi	h_hs	h_is	r1	r2	r3	r4	r5
0	0	0	0	0	1	0	0	0	0

```
age2=age**2-covamn[1]**2=63**2-70.4032= -987.6217
```

```
covs=covs[1]||age2||cos_covs[1,2:20]
```

```
covs`=
age      -7.4033
age2    -987.622
gender   0.5508
evsmoke -0.5858
sbp     -16.4142
hrx     -0.4055
diab    -0.2077
cad     -0.2492
af      -0.1519
chf     -0.1246
```

```

ic      -0.1148
murm    -0.1727
h_tia   -1.0000
h_mi    -0.0940
h_hs     0.0000
h_is    -0.0743
r1      1.0000
r2      0.0000
r3      0.0000
r4      0.0000
r5      0.0000

```

tbeta_i					
parm	tb_it	tb_im	tb_ih	tb_ii	tb_id
age	-.0020840	-0.01669	0.00000	0.11939	-0.09171
agesq	0.0000939	0.00127	0.00000	-0.00084	0.00105
gender	0.0000000	-0.00040	0.00000	0.46535	0.00000
evsmoke	0.0000000	0.00000	0.00000	0.00959	0.22035
sbp	0.0000000	0.47718	0.00511	0.00000	-0.00185
hrx	0.0000000	0.56850	0.00000	0.53176	0.00000
diab	0.0000000	0.03881	0.00000	0.63907	0.21424
cad	0.0000000	0.17361	0.00000	0.00000	-0.15475
af	0.0000000	1.51197	1.81210	0.56074	0.29047
chf	0.0000000	0.00000	0.00000	0.00000	0.14347
ic	0.0000000	0.00000	0.00000	0.00000	0.36457
murm	0.0000000	0.00000	0.00000	0.00000	0.11382
h_tia	0.0000000	-0.27400	0.00000	0.00000	0.00000
h_mi	0.0000000	0.66741	0.00000	0.00000	0.71556
h_hs	0.0000000	0.00000	0.00000	0.00000	0.00000
h_is	0.0000000	0.00000	0.00000	0.00000	0.00000
r1	0.0000000	-0.13100	-0.11330	-0.11330	-0.23900
r2	0.0000000	-0.06340	-0.06450	-0.06450	-0.13470
r3	0.0000000	0.00000	0.00000	0.00000	0.00000
r4	0.0000000	0.10010	0.10180	0.10180	0.14900
r5	0.0000000	0.13900	0.14900	0.14900	0.62390

Monthly probability of transition from is to each five possible new states:

month	tran_i				
	t_it	t_im	t_ih	t_ii	t_id
1	1	0.97993	0.99845	0.98792	0.86942
2	1	0.97450	0.99686	0.97883	0.82603
3	1	0.97043	0.99584	0.97005	0.79387
4	1	0.96754	0.99527	0.96171	0.77172
5	1	0.96560	0.99490	0.95383	0.75701
6	1	0.96437	0.99464	0.94633	0.74716
7	1	0.96366	0.99447	0.93912	0.74019
8	1	0.96326	0.99435	0.93221	0.73458
9	1	0.96297	0.99424	0.92557	0.72919
10	1	0.96261	0.99413	0.91920	0.72368
			:		
			:		
595	1	0.44309	0.92475	0.040751	7.8577E-10
596	1	0.44251	0.92465	0.040534	5.7161E-10
597	1	0.44193	0.92454	0.040317	4.0655E-10
598	1	0.44135	0.92444	0.040101	2.8273E-10
599	1	0.44077	0.92433	0.039887	1.9329E-10
600	1	0.44076	0.92432	0.039880	0

Suppose is-is, so the modified transition is:

```
tr_i=cov_beta(t(covs),tbeta_i[,5],tran_i[,5])
```

The output newevent--tr\_i(600x1):

Month	tr_I
1	0.9914568
2	0.9850125
3	0.9787677
4	0.9728188
5	0.9671865
6	0.9618097
7	0.9566371
8	0.9516593
9	0.9468714

```

10 0.9422657
    .
    .
599 0.1029059
600 0.1028929

```

## **Wait tm(end, rand):**

### **Purpose:**

Generating a randomly sampled waiting time until the next event from a survival function

### **Input requirement:**

End is the modified transition function which should be non-missing, non-increasing, bounded by 0 and 1 (including 0 and 1), one of tr\_t, tr\_h, tr\_m, tr\_i, and tr\_d whose generation is in terms of different 'old state'.

Rand is random seed value, which is 1 x 5 vector, each element is distributed as uniform (0,1), it is generated in SAS program.

### **Returns:**

Waitdays- the waiting time in days, whose survival function is the vector end.

### **Algorithm:**

The patient specific transition function is  $S(t) = S_0(t) \exp(xb)$ , where  $S_0(t)$  is the baseline function. Solving  $S(t) = S_0(t) \exp(xb) = R$ , i.e. solving  $S_0(t) = R \exp(-xb)$ , we can get the waiting time. We also assume that  $S_0(t)$  is linear during any month interval. Since  $S_0(t)$  is non-increasing, we can find the month interval that the waiting time lies by comparing  $S_0(t)$  and  $R \exp(-xb)$ , then we can get the fraction part of the waiting time by linear interpolation. The waiting time is simply set as the maximum available time if there is no enough data in  $S_0(t)$ .

---

### **SAS program:**

```

start wait_tm(end, rand);

* This counts the number of months (time intervals) in the transition
  function.;
  months=nrow(end);

* We want to use matrix operations to find the interval within which the uniform
  random variable falls. To this end, the input vector (end)
  Contains the proportion of patients surviving (without transitions) to
  the end of any particular month. The vector begin contains the corresponding
  proportion of patients beginning the interval. Begin is
  identical to end, with the exception that it begins with a 1 (i.e., all
  patients are present at the beginning of the first interval), with all
  other values shifted by 1 month.;

  begin={1} // end[1:(months-1),1];

* We identify the interval within which the uniform random variable falls as the
  only interval for which the random variable is less than or equal to the beginning
  value of the interval and greater than or equal to the ending value of the
  interval. This module has the potential to return a missing value if the transition
  function does not drop to 0 by the final interval. Also, in theory, because the
  uniform random variable is continuous there should only be one interval which
  satisfies the above condition. In order to be safe, however, I first output the
  (numbers indexing the) set of months satisfying this condition into a vector, then
  took the minimum value of this vector.;

* Deal with the special case of impossible transitions first.;
  if end[months,1]>rand then waitdays=30*months;

```

```

* Then make the calculations for the usual case;
  else do;
    temp1=loc((begin >= rand) & (rand >= end));
    interval=temp1[><];

* Not all patients will make their transition at the end of the interval.
  The following code first linearly interpolates on the scale representing
  the number of intervals, then does the same on the scale representing
  the number of days (here, we assume 30 days per month).;

    fract=(begin[interval,1]-rand)/(begin[interval,1]-end[interval,1]);
    w_months=fract+(interval-1);
    waitdays=30*w_months;
  end;

* The module returns the waiting time in days;
  return (waitdays);

finish;

```

---

### Example:

There are five possible waiting time that we might randomly select.

```

wait_t=wait_tm(tr_t,seeds[n_event_t,1]);
wait_m=wait_tm(tr_m,seeds[n_event_t,2]);
wait_h=wait_tm(tr_h,seeds[n_event_t,3]);
wait_i=wait_tm(tr_i,seeds[n_event_t,4]);
wait_d=wait_tm(tr_d,seeds[n_event_t,5]);

```

For example, we want waiting time `wait_i`, so 'end' is taken from 'newevent', i.e. `tr_i`.

Month	tr_I
1	0.9914568
2	0.9850125
3	0.9787677
4	0.9728188
5	0.9671865
6	0.9618097
7	0.9566371
8	0.9516593
9	0.9468714
10	0.9422657
.	.
.	.
599	0.1029059
600	0.1028929

Rand = seeds[n\_event\_t,4])=0.2593986

Begin:	Month	tr_i
	1	1
	2	0.9914568
	3	0.9850125
	4	0.9787677
	5	0.9728188
	6	0.9671865
	.	.
	.	.
	353	0.2609188
	354	0.2599339
	355	0.2589526

←-----rand=0.2593968

```
356 0.2579751
      .
      .
600 0.1029059
```

```
          FRACT  INTERVAL  W_MONTHS  WAITDAYS
0.5454429    355    354.54544 10636.363
```

```
fract=(begin[interval,1]-rand)/(begin[interval,1]-end[interval,1])
      = (0.2599339-0.2593986)/(0.2599339-0.2589526)=0.5455
```

```
w_months=fract+(interval-1)
          =0.54544 + (355-1)=354.54544
```

```
waitdays=30*w_months
           =10636.363 (days) .
```

### sel\_evnt(wait\_tms):

#### **Purpose:**

Selecting the smallest random time from 8 possible events

#### **Inputs:**

The inputs to the `sel_event` module are as follows:

`wait_tms` is a vector, which has 8 elements. Each element corresponds to an event and its waiting time.

- |        |                        |
|--------|------------------------|
| 1. tia | 5. death               |
| 2. mi  | 6. intervention        |
| 3. hs  | 7. patient age out     |
| 4. is  | 8. simulation time out |

#### **Input requirements:**

`wait_tms` has to be a column vector without missing value and all elements should be great than zero.

#### **Returns:**

`Ev_tm` is a vector with two elements corresponding to the event number and the time to the event.

---

#### **SAS program:**

```
start sel_evnt(wait_tms);

  event=loc(rank(wait_tms)=1);
  time=wait_tms[event,1];
  ev_tm=event||time;

  return (ev_tm);

finish;
```

---

#### **Example:**

```
wait_tms=wait_t // wait_m // wait_h // wait_i //
           wait_d // wait_int // wait_age // wait_per;
input:
wait_tms= 18000
          18000
          18000
          10636.363
          66.216713
```

99999  
36525  
18000

output: 66.216713

### Cos covs(cos covs, reg\_parm, cost tim):

#### **Purpose:**

Modifying the baseline cost function to take into account covariates

#### **Input:**

Cos\_covs is the patient characteristics minus the covariate means.

Reg\_parm is one of the beta coefficients', cbeta\_a, cbeta\_t, cbeta\_m, cbeta\_h, cbeta\_I, input data file. Each beta coefficient is 20 x 2 with the first column of parameter names not used. They are used to modify the cost s to reflect patient characteristics, of transitions from asy, tia, mi, hs, and is respectively.

Cost\_tim is the per-month cost component for each asy, tia, mi, hs, is. Each matrix has 600 x 14 column with the first column of the number month is not used.

#### **Input requirement:**

The number of columns of each cost\_asy, cost\_tia, cost\_mi, cost\_hs, and cost\_is should be equal the number of the items (columns) in the hospital cost file.

#### **Returns:**

The new\_cost is an updated corresponding matrix with dimensions of 600 x 13.

---

#### **SAS Program:**

```
start cos_covs(cos_covs, reg_parm, cos_time);  
  
    new_cost=(exp(cos_covs*reg_parm))#cos_time;  
    return (new_cost);  
  
finish;
```

---

#### **Example:**

Input:

Cos\_covs:

age	-7.403
gender	0.5508
evsmoke	-.5858
sbp	-16.41
hrx	-.4055
diab	-.2077
cad	-.2492
af	-.1519
chf	-.1246
ic	-.1148
murm	-.1727
h_tia	-1.000
h_mi	-.0940
h_hs	0.0000
h_is	-.0743
r1	1.0000
r2	0.0000
r3	0.0000
r4	0.0000
r5	0.0000

reg\_parm:

c2_beta_a	
parm	cb_asy
age	0
gender	0

evsmoke	0
sbp	0
hrx	0
diab	0
cad	0
af	0
chf	0
ic	0
murm	0
h_tia	0
h_mi	0
h_hs	0
h_is	0
r1	0
r2	0
r3	0
r4	0
r5	0

c2\_beta\_t

parm	cb_tia
age	0
gender	0
evsmoke	0
sbp	0
hrx	0
diab	0
cad	0
af	0
chf	0
ic	0
murm	0
h_tia	0
h_mi	0
h_hs	0
h_is	0
r1	0
r2	0
r3	0
r4	0
r5	0

c2\_beta\_m

parm	cb_mi
age	0
gender	0
evsmoke	0
sbp	0
hrx	0
diab	0
cad	0
af	0
chf	0
ic	0
murm	0
h_tia	0
h_mi	0
h_hs	0
h_is	0
r1	0
r2	0
r3	0
r4	0
r5	0

c2\_beta\_h

parm	cb_hs
age	0.0000
gender	0.0000
evsmoke	0.0000
sbp	0.0000
hrx	0.0000
diab	0.0000
cad	0.0000
af	0.0000
chf	0.0000
ic	0.0000
murm	0.0000
h_tia	0.0000

```

h_mi      0.0000
h_hs      0.0000
h_is      0.0000
r1        -0.6627
r2        -0.4237
r3         0.0000
r4         0.7186
r5         1.1307

```

c2\_beta\_i

```

parm      cb_is
age       0.0000
gender    0.0000
evsmoke   0.0000
sbp       0.0000
hrx       0.0000
diab      0.0000
cad       0.0000
af        0.0000
chf       0.0000
ic        0.0000
murm      0.0000
h_tia     0.0000
h_mi      0.0000
h_hs      0.0000
h_is      0.0000
r1        -0.6627
r2        -0.4237
r3         0.0000
r4         0.7186
r5         1.1307

```

In our example, reg\_parm is replaced by cbeta\_i and cost\_tim is replaced by cost\_i, here only cost\_i is printed out in the following, cost\_asy, cost\_tia, cost\_mi, and cost\_hs are not.

month	cost_is								equip	rehabu	rehabh	nh	d_non_m	hs
	acute	phy	opd	hh	snf	dme	drug							
1	1113.94	620.35	209.21	255.56	373.20	50.30	178.13	650	805.57	320.75	437.64	1085.04	6099.69	
2	645.25	359.34	121.19	148.03	216.18	29.14	103.18	0	466.63	185.79	253.50	628.51	3156.74	
3	495.67	276.04	93.09	113.72	166.06	22.38	79.26	0	358.46	142.72	194.74	482.81	2424.97	
4	431.73	240.43	81.08	99.05	144.64	19.50	69.04	0	312.21	124.31	169.61	420.53	2112.12	
5	385.13	214.48	72.33	88.36	129.03	17.39	61.59	0	278.52	110.89	151.31	375.14	1884.16	
6	339.48	189.06	63.76	77.88	113.74	15.33	54.29	0	245.51	97.75	133.37	330.68	1660.84	
7	383.52	195.64	43.36	27.04	37.03	19.66	79.95	0	12.58	12.45	271.05	408.32	1490.60	
8	381.32	194.51	43.11	26.89	36.82	19.55	79.49	0	12.51	12.37	269.49	405.98	1482.05	
9	379.12	193.39	42.86	26.73	36.61	19.44	79.03	0	12.43	12.30	267.94	403.64	1473.49	
.														
.														
599	719.56	377.53	140.92	179.58	76.94	41.45	138.51	0	24.18	12.48	470.93	1059.42	3241.49	
600	721.09	378.32	141.22	179.96	77.10	41.54	138.80	0	24.23	12.51	471.93	1061.66	3248.36	

output: newcost—cost\_i:

month	cost_is								equip	rehabu	rehabh	nh	d_non_m	hs
	acute	phy	opd	hh	snf	dme	drug							
1	574.19	319.76	107.84	131.73	192.37	25.93	91.82	335.05	415.24	165.33	225.58	559.29	3144.1	
2	332.60	185.22	62.47	76.30	111.43	15.02	53.18	0.00	240.53	95.77	130.67	323.97	1627.2	
3	255.50	142.29	47.98	58.62	85.60	11.54	40.86	0.00	184.77	73.57	100.38	248.87	1250.0	
4	222.54	123.93	41.79	51.06	74.56	10.05	35.59	0.00	160.93	64.08	87.43	216.77	1088.7	
5	198.52	110.56	37.28	45.55	66.51	8.96	31.75	0.00	143.57	57.16	77.99	193.37	971.20	
6	174.99	97.45	32.87	40.14	58.63	7.90	27.98	0.00	126.55	50.39	68.75	170.45	856.09	
7	197.69	100.84	22.35	13.94	19.09	10.13	41.21	0.00	6.48	6.42	139.71	210.47	768.34	
8	196.55	100.26	22.22	13.86	18.98	10.08	40.97	0.00	6.45	6.38	138.91	209.27	763.93	
9	195.42	99.68	22.09	13.78	18.87	10.02	40.74	0.00	6.41	6.34	138.11	208.06	759.52	
.														
.														
599	370.90	194.60	72.64	92.57	39.66	21.37	71.40	0.00	12.46	6.43	242.74	546.09	1670.9	
600	371.69	195.01	72.79	92.76	39.74	21.41	71.55	0.00	12.49	6.45	243.26	547.24	1674.4	

## Monthcost(monthvec, costs, discount, wait tim, elap tim):

### **Purpose:**

Calculating the discounted costs for each event that the patient fall into based on the waiting time and elaptime

### **Input:**

Monthvec is a vector of the month.

Costs are the modifying costs corresponding to cost\_a, cost\_t, cost\_m, cost\_h, and cost\_i.

Discount is a scalar from the cost\_dis input file. It is a monthly discount rate.

Wait\_tim is the time until the next event.

Elap\_tim is the time since the beginning of the simulation.

### **Returns:**

Monthcosts is a vector containing the month number (or numbers) that the simulate patient was accounted and each cost component that has been discounted.

### **Algorithm:**

1. Calculating the discounted costs for first month to the 600<sup>th</sup> month.
2. Translating the time until next event (wait\_tim) into months, it is the scale of the cost input matrices. The placing result is cost\_int and divided into two parts: first one is an integer portion int\_mon, second one is a remainder portion rem\_mon. The rem\_red is remainder of the value 1 for the rem\_mon. They all might be 0.
3. Translating the time until next event (elap\_tim) into months, it is the scale of the cost input matrices. The placing results are start\_int and divided into two parts: first one is an integer portion start\_mon, second one is a remainder portion start\_rem. The start\_red is remainder of the value 1 for the start\_rem. They all might be 0.
4. When calculating the monthly costs, we use two vectors as the scale of the cost input matrices vec1 and vec2, which are got from the portions of the waiting time and elaptime. The vectors costnew1 and costnew2 are discounted cost vectors, which are cut off in terms of the start point and end point of the waiting time and elaptime. Then, the scales are multiplied by costnew1 and costnew2 respectively and are summed. Based on waiting time and elaptime, there are several possible situations:
  - A. wait\_tim > 30, i.e. the waiting time is greater than one month so the scale of cost input matrix has at least 2 rows
    - i. start\_rem <= re\_red, i.e. the end point of the waiting time lies in upper of the remainder portions of the elapsed time

```
vec1=j(int_mon,1,(1-start_rem))/j(1,1,rem_mon)
vec2=j(1,1,0)/j(int_mon,1,start_rem)
costnew1=d_cost1[1:int_mon+1,]
costnew2=j(1,13,0)/d_cost1[1:int_mon,]
```
    - ii. start\_rem > re\_red, i.e. the end point of the waiting time lies in the lower of the remainder portions of the elaptime.

```
vec1=j((int_mon+1),1,(1-start_rem))/j(1,1,0)
vec2=j(1,1,0)/j(int_mon,1,start_rem)/j(1,1,(rem_mon-(1-start_rem)))
costnew1=d_cost1[1:int_mon+2,]
costnew2=j(1,13,0)/d_cost1[1:int_mon+1,]
```
  - B. wait\_tim <= 30, i.e. the waiting time is less than one month, so the scale of the cost input matrix has at most 2 rows.
    - i. cost\_int <= start\_red, i.e. the end point of the waiting time lies in the upper of the remainder portions of the elaptime

```
vec1=cost_int
costnew1=d_cost[1,]
```
    - ii. cost\_int > start\_red, i.e. the end point of the waiting time lies in the lower of the remainder portions of elaptime.

```
vect1=j(1,1,start_red)/j(1,1,0)
```

```

vect2=j(1,1,0)//j(1,1,(cost_int-start_red))
costnew1=d_cost1[1:2,]
costnew2=j(1,13,0)//d_cost1[1,]

```

In all of cases, the maximum month is 600, so the cut off month is at 600.

---

### SAS program:

```

start monthcost(monthvec, costs, discount, wait_tim, elap_tim);

d_cost1=(costs[,1]||costs[,2]||costs[,3]||costs[,4]||costs[,5]||costs[,6]||costs[,7]
||costs[,8]||costs[,9]||costs[,10]||costs[,11]||costs[,12]
||costs[,13])#
((j(nrow(costs),1,(1/(1+(discount/12)))))) ##
((j(nrow(costs),1,(elap_tim/30))+monthvec)));

cost_int=wait_tim/30;
int_mon=int(cost_int);
rem_mon=cost_int-int_mon;
rem_red=1-rem_mon;

start_int=elap_tim/30;
start_mon=int(start_int);
start_rem=start_int-start_mon;
start_red=1-start_rem;

if wait_tim > 30 then do;
  if start_rem <= rem_red then do;
    vect1=j(int_mon,1,(1-start_rem))//j(1,1,rem_mon);
    vect2=j(1,1,0)//j(int_mon,1,start_rem);
    costnew1=d_cost1[1:int_mon+1,];
    costnew2=j(1,13,0)//d_cost1[1:int_mon,];

costmon=vect1#(costnew1[,1]||costnew1[,2]||costnew1[,3]||costnew1[,4]
||costnew1[,5]||costnew1[,6]||costnew1[,7]||costnew1[,8]||costnew1[,9]
||costnew1[,10]||costnew1[,11]||costnew1[,12]||costnew1[,13])
+vect2#(costnew2[,1]||costnew2[,2]||costnew2[,3]||costnew2[,4]||
costnew2[,5]||costnew2[,6]||costnew2[,7]||costnew2[,8]||costnew2[,9]
||costnew2[,10]||costnew2[,11]||costnew2[,12]||costnew2[,13]);

*Maximum month is 600;

  if (int_mon+start_mon+1)>600 then
    costmon=costmon[1:(600-start_mon),];
  else;
costmon=costmon;
  if (int_mon+start_mon+1)>600 then
    month=monthvec[1+start_mon:600,];
  else;
    month=monthvec[1+start_mon:int_mon+start_mon+1,];

  end;

  else do;
    vect1=j((int_mon+1),1,(1-start_rem))//j(1,1,0);
    vect2=j(1,1,0)//j(int_mon,1,start_rem)//j(1,1,(rem_mon-(1-start_rem)));
    costnew1=d_cost1[1:int_mon+2,];
    costnew2=j(1,13,0)//d_cost1[1:int_mon+1,];

costmon=vect1#(costnew1[,1]||costnew1[,2]||costnew1[,3]||costnew1[,4]||
costnew1[,5]||costnew1[,6]||costnew1[,7]||costnew1[,8]||costnew1[,9]

```

```

||costnew1[,10]||costnew1[,11]||costnew1[,12]||costnew1[,13])
+vect2#(costnew2[,1]||costnew2[,2]||costnew2[,3]||costnew2[,4]||
costnew2[,5]||costnew2[,6]||costnew2[,7]||costnew2[,8]||costnew2[,9]
||costnew2[,10]||costnew2[,11]||costnew2[,12]||costnew2[,13]);

if (int_mon+start_mon+2)>600 then
  costmon=costmon[1:(600-start_mon),];
else;
costmon=costmon;
  if (int_mon+start_mon+2)>600 then
month=monthvec[1+start_mon:600,];
  else;
month=monthvec[1+start_mon:int_mon+start_mon+2,];

end;
end;

else do;
  if (cost_int <= start_red) then do;

costmon=cost_int#(d_cost1[1,1]||d_cost1[1,2]||d_cost1[1,3]||
d_cost1[1,4]||d_cost1[1,5]||d_cost1[1,6]||d_cost1[1,7]||d_cost1[1,8]
||d_cost1[1,9]||d_cost1[1,10]||d_cost1[1,11]||d_cost1[1,12]||
d_cost1[1,13]);

  if (1+start_mon)>600 then
month=monthvec[600,];
  else;
month=monthvec[1+start_mon,];

end;

else do;
vect1=j(1,1,start_red)//j(1,1,0);
vect2=j(1,1,0)//j(1,1,(cost_int-start_red));
costnew1=d_cost1[1:2,];
costnew2=j(1,13,0)//d_cost1[1,];

costmon=vect1#(costnew1[,1]||costnew1[,2]||costnew1[,3]||costnew1[,4]
||costnew1[,5]||costnew1[,6]||costnew1[,7]||costnew1[,8]||
costnew1[,9]||costnew1[,10]||costnew1[,11]||costnew1[,12]||
costnew1[,13])
+vect2#(costnew2[,1]||costnew2[,2]||costnew2[,3]||costnew2[,4]
||costnew2[,5]||costnew2[,6]||costnew2[,7]||costnew2[,8]
||costnew2[,9]||costnew2[,10]||costnew2[,11]||costnew2[,12]||
costnew2[,13]);

  if (2+start_mon)>600 then
month=monthvec[599:600,];
  else;
month=monthvec[1+start_mon:2+start_mon,];

end;
end;

*monthcosts is a matrix, maxrow(600) x 14;

monthcosts=month||costmon;
return (monthcosts);
finish;

```



acute	phy	opd	hh	snf	COSTNEW1		equip	rehabu	rehabh	nh	d_non_m	hs
					dme	drug						
3319.2	1848.4	623.37	761.48	1112.0	149.88	530.77	1936.8	2400.3	955.73	1304.0	3233.1	18175
1914.6	1066.3	359.61	439.25	641.47	86.47	306.17	0.00	1384.6	551.29	752.21	1865.0	9367.0
1464.7	815.69	275.08	336.04	490.70	66.13	234.21	0.00	1059.2	421.74	575.45	1426.7	7165.8
1270.5	707.52	238.60	291.48	425.64	57.38	203.17	0.00	918.75	365.81	499.12	1237.5	6215.4
1128.6	628.54	211.96	258.94	378.12	50.96	180.49	0.00	816.21	324.96	443.42	1099.4	5521.6
990.72	551.74	186.07	227.28	331.93	44.74	158.44	0.00	716.48	285.27	389.22	965.04	4846.9
1114.6	568.58	126.01	78.58	107.62	57.14	232.35	0.00	36.56	36.18	787.74	1186.7	4332.0
1103.6	562.95	124.77	77.82	106.56	56.58	230.06	0.00	36.21	35.80	779.95	1175.0	4289.3
1092.7	557.38	123.53	77.04	105.52	56.03	227.78	0.00	35.83	35.45	772.25	1163.4	4246.9

acute	phy	opd	hh	snf	COSTNEW2		equip	rehabu	rehabh	nh	d_non_m	hs
					dme	drug						
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3319.2	1848.4	623.37	761.48	1112.0	149.88	530.77	1936.8	2400.3	955.73	1304.0	3233.1	18175
1914.6	1066.3	359.61	439.25	641.47	86.47	306.17	0.00	1384.6	551.29	752.21	1865.0	9367.0
1464.7	815.69	275.08	336.04	490.70	66.13	234.21	0.00	1059.2	421.74	575.45	1426.7	7165.8
1270.5	707.52	238.60	291.48	425.64	57.38	203.17	0.00	918.75	365.81	499.12	1237.5	6215.4
1128.6	628.54	211.96	258.94	378.12	50.96	180.49	0.00	816.21	324.96	443.42	1099.4	5521.6
990.72	551.74	186.07	227.28	331.93	44.74	158.44	0.00	716.48	285.27	389.22	965.04	4846.9
1114.6	568.58	126.01	78.58	107.62	57.14	232.35	0.00	36.56	36.18	787.74	1186.7	4332.0
1103.6	562.95	124.77	77.82	106.56	56.58	230.06	0.00	36.21	35.80	779.95	1175.0	4289.3
1092.7	557.38	123.53	77.04	105.52	56.03	227.78	0.00	36.21	35.80	779.95	1175.0	4289.3

```
costmon=vect1#(costnew1[,1]||costnew1[,2]||costnew1[,3]||costnew1[,4]
||costnew1[,5]||costnew1[,6]||costnew1[,7]||costnew1[,8]||costnew1[,9]
||costnew1[,10]||costnew1[,11]||costnew1[,12]||costnew1[,13])
+vect2#(costnew2[,1]||costnew2[,2]||costnew2[,3]||costnew2[,4]||
costnew2[,5]||costnew2[,6]||costnew2[,7]||costnew2[,8]||costnew2[,9]
||costnew2[,10]||costnew2[,11]||costnew2[,12]||costnew2[,13])
```

acute	phy	opd	hh	snf	costmon		equip	rehabu	rehabh	nh	d_non_m	hs
					dme	drug						
2147.1	1195.7	403.24	492.58	719.33	96.95	343.34	1252.8	1552.7	618.23	843.53	2091.4	11757
2410.6	1342.5	452.75	553.04	807.63	108.86	385.48	683.93	1743.3	694.11	947.07	2348.1	12477
1623.6	904.18	304.93	372.49	543.94	73.31	259.62	0.00	1174.1	467.49	637.87	1581.5	7943.1
1339.1	745.72	251.48	307.21	448.61	60.47	214.13	0.00	968.36	385.56	526.07	1304.3	6551.0
1178.7	656.43	221.37	270.43	394.90	53.23	188.50	0.00	852.42	339.39	463.08	1148.1	5766.6
1039.4	578.86	195.22	238.46	348.24	46.94	166.22	0.00	751.70	299.29	408.36	1012.5	5085.2
1070.9	562.63	147.22	131.09	186.83	52.76	206.25	0.00	276.66	124.14	647.01	1108.4	4513.9
1107.5	564.94	125.21	78.09	106.94	56.78	230.87	0.00	36.33	35.94	782.70	1179.1	4304.4
983.89	501.88	111.23	69.37	95.01	50.45	205.10	0.00	32.27	31.92	695.35	1047.5	3824.0

```
monthcosts=month||costmon:
```

month	acute	phy	opd	hh	snf	monthcosts		equip	rehabu	rehabh	nh	d_non_m	hs
						dme	drug						
9	2147.1	1195.7	403.24	492.58	719.33	96.95	343.34	1252.8	1552.7	618.23	843.53	2091.4	11757
10	2410.6	1342.5	452.75	553.04	807.63	108.86	385.48	683.93	1743.3	694.11	947.07	2348.1	12477
11	1623.6	904.18	304.93	372.49	543.94	73.31	259.62	0.00	1174.1	467.49	637.87	1581.5	7943.1
12	1339.1	745.72	251.48	307.21	448.61	60.47	214.13	0.00	968.36	385.56	526.07	1304.3	6551.0
13	1178.7	656.43	221.37	270.43	394.90	53.23	188.50	0.00	852.42	339.39	463.08	1148.1	5766.6
14	1039.4	578.86	195.22	238.46	348.24	46.94	166.22	0.00	751.70	299.29	408.36	1012.5	5085.2
15	1070.9	562.63	147.22	131.09	186.83	52.76	206.25	0.00	276.66	124.14	647.01	1108.4	4513.9
16	1107.5	564.94	125.21	78.09	106.94	56.78	230.87	0.00	36.33	35.94	782.70	1179.1	4304.4
17	983.89	501.88	111.23	69.37	95.01	50.45	205.10	0.00	32.27	31.92	695.35	1047.5	3824.0

**Monthutils(monthvec,utilies,discount,wait tim,elap tim):**

**Purpose:**

To calculate the discounted utilities for each event that the patient fall into based on the waiting time and elaptime.

**Input:**

Monthvec is a vector of the month.

Utilies are the modified utilies corresponding to util\_a, util\_t, util\_m, util\_1, util\_2, util\_3, util\_4, util\_5.

Discount is a scalar from the util\_dis input file. It is a monthly discount rate.

Wait\_tim is the time until the next event.

Elap\_tim is the time since the beginning of the simulation.

### Returns:

Monthutils is a vector containing the month number (or numbers) that the simulate patient accounted and utilities that has been discounted.

### Algorithm:

The logic of calculating the utilities is basically same as that of calculating the costs.

1. Calculating the discounted utilities for first month to the 600<sup>th</sup> month.
2. Translating the time until next event (`wai_tim`) into months, it is the scale of the cost input matrices. The placing result is `util_int` and divided into two parts: first one is an integer portion `int_mon`, second one is a remainder portion `rem_mon`. The `rem_red` is remainder of the value 1 for the `rem_mon`. They all might be 0.
3. Translating the time until next event (`elap_tim`) into months, it is the scale of the cost input matrices. The results are placed in `start_int` and divided into two parts: the first one is an integer portion `start_mon` and the second one is a remainder portion `start_rem`. The `start_red` is remainder of the value 1 for the `start_rem`. They all might be 0.
4. When calculating the monthly utilities, we use two vectors as the scale of the cost input matrices `vec1` and `vec2`, which are taken from the portions of the waiting time and `elaptime`. The vector `costnew1` and `costnew2` are discounted cost vectors, which are cut off in terms of the start point and end point of the waiting time and `elaptime`. Then, the scales are multiplied by `costnew1` and `costnew2` respectively and are summed. Based on `waittime` (the time current event lasts) and `elaptime` (the simulation time until current state ends), there are several possible situations:
  - A. `wait_tim > 30`, i.e. the waiting time is greater than one month, so the scale of cost input matrix has at least 2 rows
    - i. `start_rem <= re_red`, i.e. the end point of the waiting time lies in upper of the remainder portions of the `elaptime`

```
vec1=j(int_mon,1,(1-start_rem))/j(1,1,rem_mon)
vec2=j(1,1,0)/j(int_mon,1,start_rem)
utilnew1=d_util1[1:int_mon+1,]
utilnew2=j(1,13,0)/d_util1[1:int_mon,]
```
    - ii. `start_rem > re_red`, i.e. the end point of the waiting time lies in the lower of the remainder portions of the `elaptime`.

```
vec1=j((int_mon+1),1,(1-start_rem))/j(1,1,0)
vec2=j(1,1,0)/j(int_mon,1,start_rem)/j(1,1,(rem_mon-(1-start_rem)))
utilnew1=d_util1[1:int_mon+2,]
utilnew2=j(1,13,0)/d_util1[1:int_mon+1,]
```
  - B. `wait_tim <= 30`, i.e. the waiting time is less than one month, so the scale of the `util` input matrix has at most 2 rows.
    - i. `util_int <= start_red`, i.e. the end point of the waiting time lies in the upper of the remainder portions of the `elaptime`.

```
vec1=util_int
utilnew1=d_util[1,]
```
    - ii. `util_int > start_red`, i.e. the end point of the waiting time lies in the lower of the remainder portions of `elaptime`.

```
vect1=j(1,1,start_red)/j(1,1,0)
vect2=j(1,1,0)/j(1,1,(util_int-start_red))
utilnew1=d_util1[1:2,]
utilnew2=j(1,13,0)/d_util1[1,]
```

In all of cases, the maximum month is 600, so the cut off month is at 600.

### SAS program:

---

```

start monthutils(monthvec,utils,util_dis,wait_tim,elap_tim);

d_util1=utils #
  ((j(nrow(utils),1,(1/(1+(util_dis/12)))))) ##
  ((j(nrow(utils),1,(elap_tim/30))+monthvec));

cost_int=wait_tim/30;
int_mon=int(cost_int);
rem_mon=cost_int-int_mon;
rem_red=1-rem_mon;

end_int=(elap_tim+wait_tim)/30;
end_mon=int(end_int);
end_rem=end_int-end_mon;

start_int=elap_tim/30;
start_mon=int(start_int);
start_rem=start_int-start_mon;
start_red=1-start_rem;

if wait_tim > 30 then do;
  if start_rem <= rem_red then do;
    vect1=j(int_mon,1,(1-start_rem))/j(1,1,rem_mon);
    vect2=j(1,1,0)/j(int_mon,1,start_rem);
    utilnew1=d_util1[1:int_mon+1,];
    utilnew2=j(1,1,0)/d_util1[1:int_mon,];
    utilmon=vect1#utilnew1+vect2#utilnew2;
    if (int_mon+start_mon+1)>600 then
      utilmon=utilmon[1:(600-start_mon)];
    else; utilmon=utilmon;
    if (int_mon+start_mon+1)>600 then
      month=monthvec[1+start_mon:600];
    else;
    month=monthvec[1+start_mon:int_mon+start_mon+1,];

  end;

  if start_rem > rem_red then do;
    vect1=j((int_mon+1),1,(1-start_rem))/j(1,1,0);
    vect2=j(1,1,0)/j(int_mon,1,start_rem)/j(1,1,(rem_mon-(1-start_rem)));
    utilnew1=d_util1[1:int_mon+2,];
    utilnew2=j(1,1,0)/d_util1[1:int_mon+1,];
    utilmon=vect1#utilnew1+vect2#utilnew2;
    if (int_mon+start_mon+2)>600 then
      utilmon=utilmon[1:(600-start_mon)];
    else;
    utilmon=utilmon;
    if (int_mon+start_mon+2)>600 then
      month=monthvec[1+start_mon:600];
    else;
    month=monthvec[1+start_mon:int_mon+start_mon+2,];

  end;
end;

else do;
  if (cost_int <= start_red) then do;
    utilmon=cost_int#d_util1[1,1];
    if (1+start_mon)>600 then
      month=monthvec[600,];
  end;
end;

```

```

else;
month=monthvec[1+start_mon,];
*print 'Waittime less than 30 days and cost_int < start_red';
*print month utilmon;
end;

if (cost_int>start_red) then do;
vect1=j(1,1,start_red)//j(1,1,0);
vect2=j(1,1,0)//j(1,1,(cost_int-start_red));
utilnew1=d_util1[1:2,];
utilnew2=j(1,1,0)//d_util1[1,1];
utilmon=vect1#utilnew1+vect2#utilnew2;

if (2+start_mon)>600 then
month=monthvec[599:600,];
else;
month=monthvec[1+start_mon:2+start_mon,];

end;
end;
monthutils=month||utilmon;
return (monthutils);
finish;

```

---

**Example:**

Input:

Monthvect={1,2,3.....,600}

Utils:

month	UTIL_a	UTIL_t	UTIL_MM	UTIL_1	UTIL_2	UTIL_3	UTIL_4	UTIL_5
1	0.0833333	0.075	0.0708333	0.0666667	0.0541667	0.0416667	0.0291667	0.0166667
2	0.0833333	0.075	0.0708333	0.0666667	0.0541667	0.0416667	0.0291667	0.0166667
3	0.0833333	0.075	0.0708333	0.0666667	0.0541667	0.0416667	0.0291667	0.0166667
4	0.0833333	0.075	0.0708333	0.0666667	0.0541667	0.0416667	0.0291667	0.0166667
5	0.0833333	0.075	0.0708333	0.0666667	0.0541667	0.0416667	0.0291667	0.0166667
.	.	.	.	.	.	.	.	.
599	0.0833333	0.075	0.0708333	0.0666667	0.0541667	0.0416667	0.0291667	0.0166667
600	0.0833333	0.075	0.0708333	0.0666667	0.0541667	0.0416667	0.0291667	0.0166667
				WAIT_TIM	ELAP_TIM			
				256.31322	250.59382			

util\_dis=0.5

```

d_util1=utils #
((j(nrow(utils),1,(1/(1+(util_dis/12)))))) ##
((j(nrow(utils),1,(elap_tim/30))+monthvec))

month D_UTIL1
1 0.0160309
2 0.0159644
3 0.0158982
4 0.0158322
5 0.0157665
.
599 0.0013338
600 0.0013283

```

In the above case, waittime>30 and start\_rem <= rem\_red

```
vect1=j(int_mon,1,(1-start_rem))//j(1,1,rem_mon)
```

```

vect2=j(1,1,0)//j(int_mon,1,start_rem)
utilnew1=d_util1[1:int_mon+1,]
utilnew2=j(1,1,0)//d_util1[1:int_mon,]

```

```

      VECT1      VECT2
0.6468726      0
0.6468726 0.3531274
0.6468726 0.3531274
0.6468726 0.3531274
0.6468726 0.3531274
0.6468726 0.3531274
0.6468726 0.3531274
0.6468726 0.3531274
0.5437742 0.3531274

      UTILNEW1  UTILNEW2
0.0160309      0
0.0159644 0.0160309
0.0158982 0.0159644
0.0158322 0.0158982
0.0157665 0.0158322
0.0157011 0.0157665
0.0156359 0.0157011
0.0155711 0.0156359
0.0155065 0.0155711

```

```

utilmon=vect1#utilnew1+vect2#utilnew2 :

```

```

      UTILMON
      0.01037
      0.0159879
      0.0159216
      0.0158555
      0.0157897
      0.0157242
      0.0156589
      0.015594
      0.0139306

```

```

monthutils=month||utilmon:

```

```

      month      UTILMON
      9          0.01037
      10         0.0159879
      11         0.0159216
      12         0.0158555
      13         0.0157897
      14         0.0157242
      15         0.0156589
      16         0.015594
      17         0.0139306

```

All of examples above are run from small samples, because it would take too much time at each run if using 10,000 simulating patients.