# Analysis of Model F

*Hongshan*

*September 2017*

## Contents

### Analysis on Model F

```r
suppressMessages(source("ams_initialize_script.R"))
suppressMessages(library(RxODE))
suppressMessages(library(dplyr))
```

## Model F is defined below

```r
ivsc_4cmtct_shedct = function() {
  model           = list()
  model$name      = as.character(sys.calls()[[sys.nframe()]])

  #COMPARTMENTS
  model$cmtshort  = c('AmtD0','D1','D2','D3','S1','S3','M3','DS1','DS3','DM3','DM1','M1')
  #CALCULATE INITIAL CONDITION WITH NO DRUG PRESENT AND ASSUMING STEADY STATE
  model$init      = function(p){
              init = c(AmtD0=0,D1=0,D2=0,D3=0,S1=0,S3=0,M3=0,DS1=0,DS3=0,DM3=0,DM1=0,M1=0)
              p    = p %>% t() %>% as.data.frame()


              ksyn = with(p,c(ksynS1,ksynS3,ksynM1,ksynM3))
              K    = with(p,matrix(c( keS1+k13S    , -k31S*VS3/VS1       , -kshedM1           ,  0,
                                   -k13S*VS1/VS3 ,  keS3+k31S          ,  0                 , -kshedM3
                                    0            ,  0                  ,  kshedM1+keM1+k13M , -k31M*VD
                                    0            ,  0                  , -k13M*VD1/VD3      ,  kshedM3
                                  nrow = 4, byrow=TRUE))
              x    = solve(K,ksyn)

              init["S1"] = unlist(x[1])
```

```
            init["S3"] = unlist(x[2])
            init["M1"] = unlist(x[3])
            init["M3"] = unlist(x[4])
            return(init)
  }

  #PARAMEETRS IN MODEL
  model$pin       = c('F','ka','VD1','VD2','VD3','VS1','VS3','VDS1','VDS3',
                      'k12D','k21D','k13D','k31D','k13S','k31S','k13DS','k31DS',
                      'ksynS1','ksynS3','ksynM3','keD1','keD3','keS1','keS3','keDS1','keDS3','keM3','ke
                      'kon1','koff1','kon3','koff3',
                      'kshedM3','kshedDM3','ksynM1','kshedM1','kshedDM1','keM1','keDM1','k13M','k31M','
  model$pode      = model$pin

                  #INPUT/SYNTHESIS/SHED   DISTRIBUTION (CENTRAL/TUMOR)      BINDING
  model$rxode.str = '
     D1            = AmtD1/VD1;
     d/dt(AmtD0)   =  -ka *AmtD0;
     d/dt(AmtD1)   =(F*ka *AmtD0/VD1 - k13D *D1 +  k31D *VD3/VD1*D3  - keD1 *D1  - kon1*D1*S1 + koff1*DS
     d/dt(D2)      = k12D*VD1/VD2*D1 - k21D*D2;
     d/dt(D3)      = k13D *VD1/VD3*D1 - k31D*D3  - keD3 *D3  - kon3*D3*(S3+M3) + koff3*(DS3+DM3);
     d/dt(S1)      = ksynS1+kshedM1*M1 - k13S *S1 +  k31S*VS3/VS1*S3  - keS1 *S1  - kon1*D1*S1      + ko
     d/dt(S3)      = ksynS3 +kshedM3*M3   + k13S *VS1/VS3*S1 - k31S*S3  - keS3 *S3  - kon3*D3*S3  + koff
     d/dt(M3)      = ksynM3 -kshedM3*M3  -k31M*M3+k13M*VD1/VD3*M1  - keM3 *M3  - kon3*D3*M3 + koff3*DM3;
     d/dt(DS1)     = kshedM1*DM1 - k13DS*DS1 + k31DS*VDS3/VDS1*DS3 - keDS1*DS1 + kon1*D1*S1 - koff1*DS1;
     d/dt(DS3)     = kshedDM3*DM3 + k13DS*VDS1/VDS3*DS1 - k31DS*DS3 - keDS3*DS3 + kon3*D3*S3  - koff3*DS
     d/dt(DM3)     = -kshedDM3*DM3  - keDM3*DM3 + kon3*D3*M3 - koff3*DM3-k31DM*DM3+k13DM*(VD1/VD3)*DM1;
     d/dt(DM1)     = -keDM1*DM1 -kshedDM1*DM1 +kon1*D1*M1 -koff1*DM1-k13DM*DM1+k31DM*(VD3/VD1)*DM3;
     d/dt(M1)      = ksynM1 -kshedM1*M1 -keM1*M1 +k31M*VD3/VD1*M3 -k13M*M1 -kon1*D1*M1 +koff1*DM1;
  '
  model$rxode     = RxODE(model = model$rxode.str, modName = model$name)

  model$rxout     = function(result)    {
    result        = as.data.frame(result)
    result = mutate(result,
                    Dtot1 = D1+DS1,
                    Stot1 = S1+DS1,
                    Dtot3 = D3+DS3,
                    Stot3 = S3+DS3,
                    Mtot1 = M1+DM1,
                    Mtot3 = M3+DM3)
  }

  return(model)
}
```

```
# Global Variables
model = ivsc_4cmtct_shedct()
tmax = 3*28 # End of the observation time, unit=day
tau = 14 # dosing interval, unit=day
compartment = 2 # compartment to which dosing is applied

# Import parameters
d = readxl::read_excel("../data/ivsc_4cmtct_shedct_param.xlsx",1)
```

```r
param.as.double = d$Value
names(param.as.double) = d$Parameter

# Helper function to make the range of a variable when performing sensitivity analysis
lseq = function(from, to, length.out){
    sequence = seq(log(from), log(to), length.out=length.out)
    sequence = exp(sequence)
    return(sequence)
}
```

```r
simulation = function(dose.nmol, params_file_path, tmax){
  d <- xlsx::read.xlsx(params_file_path, 1)
  param.as.double = d$Value
  names(param.as.double) = d$Parameter
  ev = eventTable(amount.units="nmol", time.units="days")
  sample.points = c(seq(-7, tmax, 0.1), 10^(-3:0)) # sample time, increment by 0.1
  sample.points = sort(sample.points)
  sample.points = unique(sample.points)
  ev$add.sampling(sample.points)
  ev$add.dosing(dose=dose.nmol, nbr.doses=floor(tmax/tau)+1, dosing.interval=tau,
                dosing.to=2)

  init = model$init(param.as.double)
  out = model$rxode$solve(param.as.double, ev, init)
  out = model$rxout(out)
  out = out %>%
    mutate(Sfree.pct = S1/init["S1"],
           Mfree.pct = M3/init["M3"],
           dose.nmol = dose.nmol)
  return(out)
}
```

## Test the theory and simulation of the lumped parameters: M30, M3tot.ss, B, Davg3

### The function below takes a dataset and calculate these lumped parameters from theory

```r
lumped.parameters.theory = function(params_file_path, dose.nmol){
    d <- xlsx::read.xlsx(params_file_path, 1)
    param.as.double = d$Value
    names(param.as.double) = d$Parameter
    p = as.data.frame(t(param.as.double))
    Kss = with(p, (koff3 + keDM3 + kshedM3)/kon3)
    Kd = with(p, koff3 / kon3)

    # numerators for M3.0 and Mtot3.ss(Mtot3 at steady state, M3 at initial state)
    numerator.DM   = with(p, k13DM*(VD1/VD3)*ksynM1+(keDM1+kshedDM1+k13DM)*ksynM3)
    denomenator.DM = with(p, (keDM1+kshedDM1+k13DM)*(keDM3+kshedM3+k31DM)-k31DM*k13DM)

    numerator.M    = with(p, k13M *(VD1/VD3)*ksynM1+(keM1 +kshedM1 +k13M) *ksynM3)
    denomenator.M  = with(p, (keM1 +kshedM1 +k13M) *(keM3 +kshedM3+k31M) -k31M *k13M)

    # numerator and denomenator for M3.0 ()
    Mtot3.ss = numerator.DM / denomenator.DM
    M30      = numerator.M  / denomenator.M

    # Target accumulation in the tumor compartment
    Tacc.tum = Mtot3.ss / M30

    # Biodistribution coefficient (reference: ModelF_Appendix)
    B = with(p, (k13D/(keD3 + k31D) * (VD1/VD3)))

    # Clearance
    CL = with(p, (keD1 / VD1))

    # Average drug concentration in the central compartment
    # Cavg1 = with(p, (F * dose.nmol) / (CL * tau))


    # Average drug concentratio in the tumor compartment (I have no idea how to compute it)


    # AFIRT computed with Kss
    AFIRT.theory.Kss = Kss*Tacc.tum*(CL*tau)/(dose.nmol*B)

    # AFIRT computed with Kd
    AFIRT.theory.Kd  = Kd*Tacc.tum*(CL*tau)/(dose.nmol*B)


    lumped_parameters_theory = data.frame(type = "theory",
                                          M30=M30,
                                          Mtot3.ss=Mtot3.ss,
                                          Tacc.tum=Tacc.tum,
                                          B = B,
                                          CL = CL,
                                          AFIRT.Kss = AFIRT.theory.Kss,
                                          AFIRT.Kd = AFIRT.theory.Kd)



    return(lumped_parameters_theory)
}
```

### Calculate lumped parameters from theory for Atezolizumab

```
lumped_parameters_theory = lumped.parameters.theory("../data/ModelF_Atezolizumab_Params.xlsx", 80)
print(lumped_parameters_theory)
```

```
##     type      M30 Mtot3.ss Tacc.tum         B         CL  AFIRT.Kss
## 1 theory 2.551332 2.551332        1 0.3333333 0.01859012 0.04879908
##     AFIRT.Kd
## 1 0.02439954
```

Okay, for different inital doses, we do get the same simulated lumped parameters. However, M30 seems to be the same as Mtot3.ss in the simulation which is kind of weird. Also, M30 calculated from theory and M30 calculated from simulation are off by a lot.

- Andy says: the parameetrs are such that M3tot doesn't accumulate much. So this makes sense. We compare the theory and simulation below.
- kable is not working for lumped_parameters_sim or lumped_parameters_theory.
- Hongshan says: AFIRT.sim and AFIRT.theory still differ by a big margin

### Sensitivity analysis on lumped parameters calculated from theory with respect to dose.mol

```
dose.nmol.range = lseq(1, 1000, 10)
table = data.frame()
for (dose.nmol in dose.nmol.range){
  row = lumped.parameters.theory("../data/ModelF_Atezolizumab_Params.xlsx", dose.nmol)
  table = rbind(table, row)
}
print(table)
```

```
##      type      M30 Mtot3.ss Tacc.tum         B         CL   AFIRT.Kss
## 1  theory 2.551332 2.551332        1 0.3333333 0.01859012 3.903926234
## 2  theory 2.551332 2.551332        1 0.3333333 0.01859012 1.812042042
## 3  theory 2.551332 2.551332        1 0.3333333 0.01859012 0.841075411
## 4  theory 2.551332 2.551332        1 0.3333333 0.01859012 0.390392623
## 5  theory 2.551332 2.551332        1 0.3333333 0.01859012 0.181204204
## 6  theory 2.551332 2.551332        1 0.3333333 0.01859012 0.084107541
## 7  theory 2.551332 2.551332        1 0.3333333 0.01859012 0.039039262
## 8  theory 2.551332 2.551332        1 0.3333333 0.01859012 0.018120420
## 9  theory 2.551332 2.551332        1 0.3333333 0.01859012 0.008410754
## 10 theory 2.551332 2.551332        1 0.3333333 0.01859012 0.003903926
##       AFIRT.Kd
## 1  1.951963117
## 2  0.906021021
## 3  0.420537705
## 4  0.195196312
## 5  0.090602102
## 6  0.042053771
## 7  0.019519631
## 8  0.009060210
## 9  0.004205377
## 10 0.001951963
```

### The function below simulates the lumped parameter

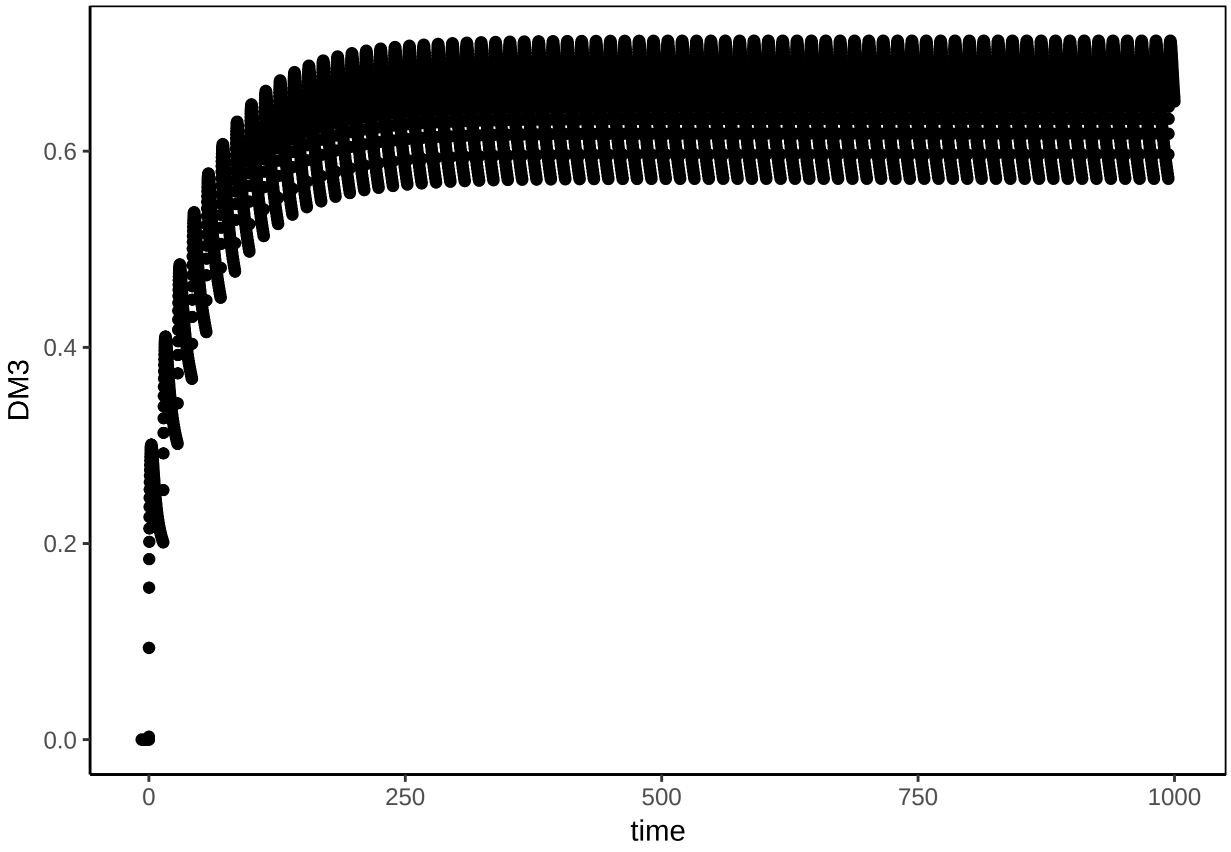
```
# Change the function <simulation> so that it includes more time points

simulation = function(dose.nmol, params_file_path, tmax){
  d <- xlsx::read.xlsx(params_file_path, 1)
  param.as.double = d$Value
  names(param.as.double) = d$Parameter
  ev = eventTable(amount.units="nmol", time.units="days")
  sample.points = c(seq(-7, tmax, 0.1), 10^(-3:0)) # sample time, increment by 0.1
  sample.points = sort(sample.points)
  sample.points = unique(sample.points)
  ev$add.sampling(sample.points)
  ev$add.dosing(dose=dose.nmol, nbr.doses=floor(tmax/tau)+1, dosing.interval=tau,
                dosing.to=2)

  init = model$init(param.as.double)
  out = model$rxode$solve(param.as.double, ev, init)
  out = model$rxout(out)
  out = out %>%
    mutate(Sfree.pct = S1/init["S1"],
           Mfree.pct = M3/init["M3"],
           dose.nmol = dose.nmol)
  return(out)
}
```

**Run the simulation longer to make sure the system reaches steady state**

```
sim = simulation(dose.nmol = 80, params_file_path ="../data/ModelF_Atezolizumab_Params.xlsx", tmax=1000
g = ggplot(sim, aes(x=time, y=DM3)) + geom_point()
print(g)
```



## It is quite visible that the system reaches steady state after time > 500.

## Simulate lumped parameters during steady state

```r
lumped.parameters.simulation = function(params_file_path, dose.nmol, tmax){
  sim = simulation(dose.nmol=dose.nmol, params_file_path = params_file_path, tmax=tmax)
  initial_state = sim %>%
    filter(time==0)
  M30 = initial_state$M3

  steady_state = sim %>%
    filter(time>(tmax/2) & time <tmax)
  Mtot3.ss = mean(steady_state$Mtot3)

  Tacc.tum = Mtot3.ss / M30

  # Average drug concentration in central compartment
  dose_applied = sim %>%
    filter(time > 0)
  Cavg1 = mean(dose_applied$D1)


  # Average drug concentration in tumor compartment
  Cavg3 = mean(dose_applied$D3)

  # AFIRT
  AFIRT.sim = mean(steady_state$Mfree.pct)



  lumped_parameters_sim = data.frame(type = "simulation",
                                     M30=M30,
                                     Mtot3.ss=Mtot3.ss,
                                     Tacc.tum=Tacc.tum,
                                     Cavg1 = Cavg1,
                                     Cavg3 = Cavg3,
                                     AFIRT.sim = AFIRT.sim)

  return(lumped_parameters_sim)
}


lump_sim = lumped.parameters.simulation(params_file_path="../data/ModelF_Atezolizumab_Params.xlsx", dos
lump_sim
```

```
##         type      M30 Mtot3.ss Tacc.tum    Cavg1    Cavg3 AFIRT.sim
## 1 simulation 2.551332 2.551332        1 24.93032 2.642064 0.7496819
```

## Sensitivity analysis on lumped parameters calculated from simulation with respect to dose.nmol

```r
dose.nmol.range = lseq(1, 1000, 10)
table = data.frame()
for (dose.nmol in dose.nmol.range){
  row = lumped.parameters.simulation("../data/ModelF_Atezolizumab_Params.xlsx", dose.nmol, tmax=1000)
  table = rbind(table, row)
}
print(table)
```

```
##          type      M30 Mtot3.ss Tacc.tum       Cavg1       Cavg3 AFIRT.sim
## 1  simulation 2.551332 2.551332        1   0.3028735  0.04281261 0.9920676
## 2  simulation 2.551332 2.551332        1   0.6531009  0.09064673 0.9836560
## 3  simulation 2.551332 2.551332        1   1.4095540  0.18908895 0.9676557
## 4  simulation 2.551332 2.551332        1   3.0467562  0.38664935 0.9397146
## 5  simulation 2.551332 2.551332        1   6.6000877  0.77845671 0.8947114
## 6  simulation 2.551332 2.551332        1  14.3403694  1.57375399 0.8240241
## 7  simulation 2.551332 2.551332        1  31.2996882  3.29495763 0.7113479
## 8  simulation 2.551332 2.551332        1  68.7658355  7.56145887 0.5361750
## 9  simulation 2.551332 2.551332        1 151.9242686 21.35253450 0.3032989
## 10 simulation 2.551332 2.551332        1 335.3729708 69.90695790 0.1206876
```

```
lumped.parameters.theory("../data/ModelF_Atezolizumab_Params.xlsx", 80)
```

```
##     type      M30 Mtot3.ss Tacc.tum         B         CL  AFIRT.Kss
## 1 theory 2.551332 2.551332        1 0.3333333 0.01859012 0.04879908
##     AFIRT.Kd
## 1 0.02439954
```

## under high dose, AFIRT.thy shoudl agree with AFIRT.sim

## Make a plot to demonstrate that

### Make data frames for lumped parameters(thy and sim) at different doses

```
dose.nmol.range = lseq(80, 8000, 50)

df_sim = data.frame() # put all simulations for different dose into one data frame
for (dose.nmol in dose.nmol.range){
  row = lumped.parameters.simulation("../data/ModelF_Atezolizumab_Params.xlsx", dose.nmol, tmax=100)
  df_sim = rbind(df_sim, row)
}

df_thy = data.frame() # put all theoretical calculations of lumped parameters at different dose togethe
for (dose.nmol in dose.nmol.range){
  row = lumped.parameters.theory("../data/ModelF_Atezolizumab_Params.xlsx", dose.nmol)
  df_thy = rbind(df_thy, row)
}
```

### Get AFIRT from both df_sim and df_thy

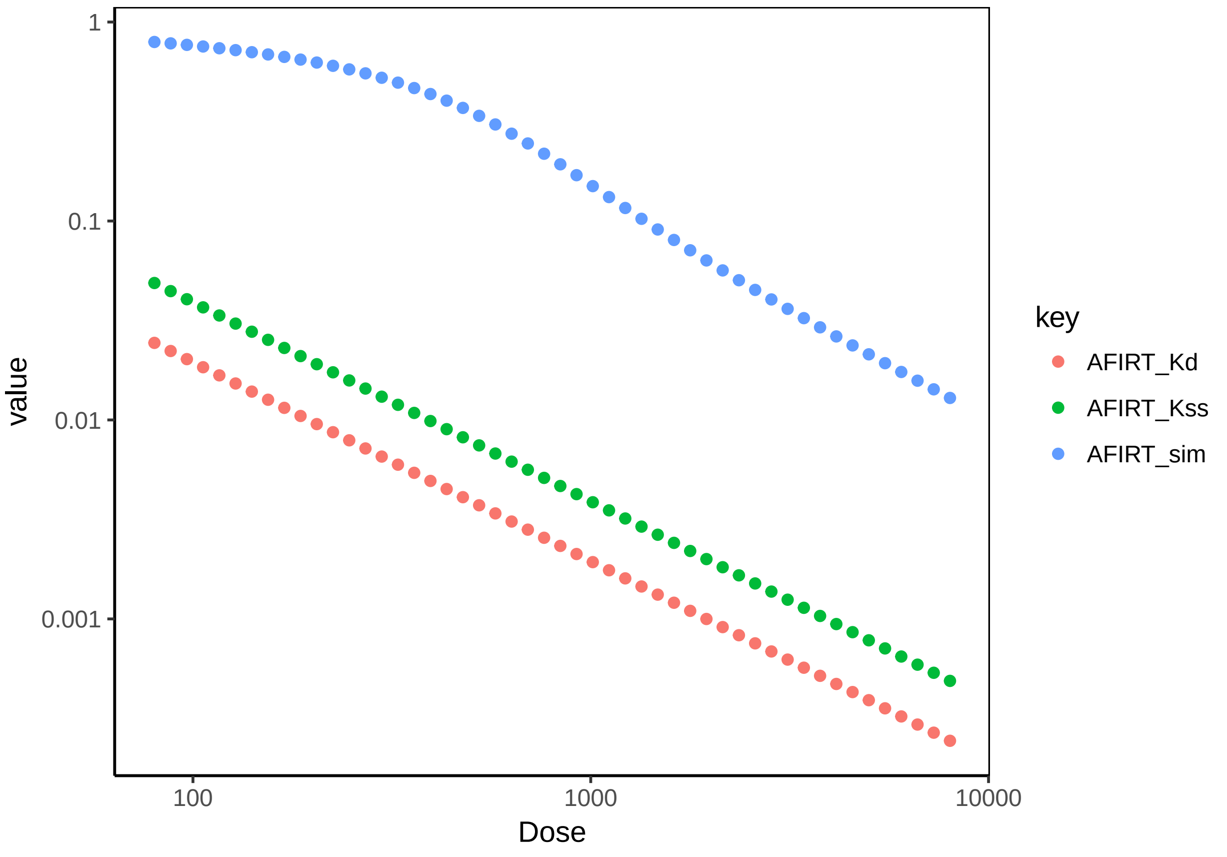
```
Dose = as.data.frame(dose.nmol.range)
AFIRT_sim = as.data.frame(df_sim$AFIRT.sim)
AFIRT_Kd= as.data.frame(df_thy$AFIRT.Kd)
AFIRT_Kss = as.data.frame(df_thy$AFIRT.Kss)
```

```r
AFIRT = data.frame(Dose, AFIRT_Kd, AFIRT_Kss, AFIRT_sim)
names(AFIRT)= c("Dose","AFIRT_Kd", "AFIRT_Kss", "AFIRT_sim")
AFIRT
```

```
##          Dose     AFIRT_Kd    AFIRT_Kss  AFIRT_sim
## 1    80.00000 0.0243995390 0.0487990779 0.79395861
## 2    87.88329 0.0222108559 0.0444217117 0.78126653
## 3    96.54341 0.0202185016 0.0404370032 0.76781694
## 4   106.05691 0.0184048652 0.0368097304 0.75355757
## 5   116.50788 0.0167539152 0.0335078305 0.73843168
## 6   127.98870 0.0152510585 0.0305021171 0.72237819
## 7   140.60085 0.0138830108 0.0277660216 0.70533125
## 8   154.45582 0.0126376794 0.0252753589 0.68722042
## 9   169.67607 0.0115040566 0.0230081131 0.66797153
## 10  186.39614 0.0104721217 0.0209442434 0.64750894
## 11  204.76383 0.0095327533 0.0190655066 0.62575779
## 12  224.94150 0.0086776480 0.0173552960 0.60264958
## 13  247.10749 0.0078992471 0.0157984943 0.57812866
## 14  271.45774 0.0071906703 0.0143813406 0.55216164
## 15  298.20750 0.0065456541 0.0130913081 0.52474917
## 16  327.59320 0.0059584970 0.0119169939 0.49593997
## 17  359.87461 0.0054240089 0.0108480179 0.46584518
## 18  395.33707 0.0049374654 0.0098749309 0.43465285
## 19  434.29404 0.0044945658 0.0089891316 0.40263768
## 20  477.08987 0.0040913951 0.0081827901 0.37016414
## 21  524.10285 0.0037243895 0.0074487789 0.33767899
## 22  575.74854 0.0033903049 0.0067806099 0.30568932
## 23  632.48346 0.0030861884 0.0061723768 0.27472765
## 24  694.80910 0.0028093517 0.0056187034 0.24530350
## 25  763.27638 0.0025573477 0.0051146955 0.21785072
## 26  838.49051 0.0023279490 0.0046558980 0.19268251
## 27  921.11632 0.0021191277 0.0042382554 0.16996631
## 28 1011.88417 0.0019290381 0.0038580762 0.14972660
## 29 1111.59640 0.0017559999 0.0035119997 0.13186932
## 30 1221.13437 0.0015984835 0.0031969670 0.11622002
## 31 1341.46635 0.0014550966 0.0029101932 0.10256076
## 32 1473.65598 0.0013245718 0.0026491436 0.09066023
## 33 1618.87172 0.0012057553 0.0024115106 0.08029380
## 34 1778.39719 0.0010975968 0.0021951937 0.07125484
## 35 1953.64248 0.0009991404 0.0019982808 0.06335961
## 36 2146.15664 0.0009095157 0.0018190314 0.05644806
## 37 2357.64136 0.0008279305 0.0016558609 0.05038281
## 38 2589.96603 0.0007536636 0.0015073272 0.04504660
## 39 2845.18424 0.0006860586 0.0013721172 0.04033989
## 40 3125.55195 0.0006245179 0.0012490358 0.03617813
## 41 3433.54741 0.0005684975 0.0011369950 0.03248953
## 42 3771.89309 0.0005175022 0.0010350045 0.02921304
## 43 4143.57974 0.0004710813 0.0009421627 0.02629653
## 44 4551.89282 0.0004288245 0.0008576490 0.02369547
## 45 5000.44154 0.0003903582 0.0007807163 0.02137159
## 46 5493.19076 0.0003553423 0.0007106846 0.01929194
## 47 6034.49605 0.0003234675 0.0006469349 0.01742806
## 48 6629.14218 0.0002944518 0.0005889037 0.01575526
## 49 7282.38542 0.0002680390 0.0005360779 0.01425204
## 50 8000.00000 0.0002439954 0.0004879908 0.01289967
```

Hongshan says: We don't actually need a plot to realize that the theory and simulation differ by a big margin But I will make a plot anyway, just to see which one is linear

```r
names = names(AFIRT)
data = AFIRT %>% gather(key, value, -c(get(names[1])))
g = ggplot(data, aes(Dose, value, color=key)) +
  scale.x.log10() +
  scale.y.log10() +
  geom_point()
g
```



Hongshan says: Deeply confused