**Six Degrees of Kevin Bacon**

Bacon number for an actor is calculated by the degree of association of the actor with Kevin Bacon.

For example: Using Jon Lovitz as the actor, we want information as follows:

Jon Lovitz ( (was in) Movie X ( (with) Actor2 ((who was in) Movie Y ( (with) Kevin Bacon.

So for that we need to download the list of all the actors with their details from the IMDB website and spray in the THOR cluster.

After that we need to perform series of operations as indicated below

So, in the below sections you will come to understand on what needs to be done with the data and how it needs to be coded on R. So the first lined of the code is

*hpcc.begin(import=’Std’)*

1. **Processing the Data : Extract, *Transform, and Load***

Now we are having the file *FileActors* which contains all the records

In this section, we will write code to transform the original actor data as follows:

1. From the raw actors’ data, we will do an ETL operation (Extract, Transform and Load) to build an actor\_movie relation set.
2. We will also construct a Kevin Bacon degrees of separation lookup set.

This is the structure we will query to answer the question:

How many degrees of separation exist between Actor X and Kevin Bacon?

1. Filter out the movie records we do not want in building our Kevin Bacon Number sets.
2. Produce a slimmed down version of the IMDB actor AND actress files to permit more efficient join operations.

*#for filtering records*

*#we need to eliminate records of ‘Boffo ’ and ‘Slasher’. For that we need to filter out records whose movies names start with ‘Boffo’ and ‘Slasher Film’*

*hpcc.string.find('condn1','IMDB.FileActors.moviename','Boffo',reqNumber=1,output=0)*

*hpcc.string.find('condn2','IMDB.FileActors.moviename','Slasher Film',reqNumber=1,output=0)*

*# Below two lines create conditions on the Boffo and Slasher Names. The specify that the names should not start with Boffo and Slasher Filmrespectively*

*condn1 <- condition(field='condn1',operator='=',keyword=0)*

*condn2 <- condition(field='condn2',operator='=',keyword=0)*

*#filter conditions to check actor name and movie type are not EMPTY strings*

*condn3 <- condition(field='actorname',operator='!=','',type='STRING')*

*condn4 <- condition(field='movie\_type',operator='!=',"Video",type='STRING')*

*# condition to filter out TV Series and Promotionals for TV*

*condn5 <- condition(field='isTVSeries',operator='=','N',type='STRING')*

*condn6 <- condition(field='movie\_type',operator='!=',"For TV",type='STRING')*

*#Joining all the above conditions for keeping them in the filter statment*

*cond <- condition(field=condn1,operator='AND',keyword=condn2,type='NUM')*

*cond <- condition(field=cond,operator='AND',keyword=condn3,type='NUM')*

*cond <- condition(field=cond,operator='AND',keyword=condn4,type='NUM')*

*cond <- condition(field=cond,operator='AND',keyword=condn5,type='NUM')*

*cond <- condition(field=cond,operator='AND',keyword=condn6,type='NUM')*

*#filtering Records*

*hpcc.filter(data='IMDB.FileActors',condition=cond,out.dataframe='ds\_IMDB',output=0)*

*#We need only two fields actor name and movie names. So we slim down the records*

*#For that we need to specify the structure we need to have*

*hpcc.define.record('slim\_IMDB\_rec',c('STRING','STRING150'),c('actor','movie'))*

*# we need to specify a transform function to specify on how to slim down the records*

*xd <- hpcc.string.replace(out.dataframe='SELF.actor','L.actorname','(I)','',submit=FALSE)*

*hpcc.define.transform(returnType='slim\_IMDB\_rec',transformName='slim\_it',argTypes='ds\_IMDB',argNames='L',xd,'SELF.movie := L.moviename;')*

*#Finally slimming the records*

*hpcc.project(dataframe='ds\_IMDB',calltransfunc='slim\_it',out.dataframe='ActorsInMovies',output=0)*

**Links and Degree of Separation**

Now that we have our data in a useful format, have a relation defined, and the file is in place, we can write code to use the new data file.

We want to know how many actors are at a distance of N from Kevin Bacon. To accomplish this, we will construct sets of Kevin Bacon's costars that are N number apart.

We will build code which counts the number of actors with "bacon numbers" starting from 1 thru 7 that is up to 7 Levels of separation.

So, the first step will be to obtain all the movies of Kevin Bacon

*hpcc.string.find(out.dataframe='cdnsECL1','ActorsInMovies.actor','Kevin',reqNumber=1,output=0)*

*cdn1 <- condition('cdnsECL1',operator='>',0)*

#condition to check if ‘Bacon’ exists in the Actor’s name

*hpcc.string.find(out.dataframe='cdnsECL2','ActorsInMovies.actor','Bacon',reqNumber=1,output=0)*

*cdn2 <- condition('cdnsECL2',operator='>',0)*

*hpcc.filter(out.dataframe='AllKBEntries',condition=condition(cdn1,operator='AND',keyword=cdn2,type='NUM'),data='ActorsInMovies',output=0)*

*#Now deleting all the duplicate Records*

*hpcc.dedup(output=0,dataframe='AllKBEntries',out.dataframe='KBMovies',condition='movie',all='ALL')*

*#Now we have all the Kevin Bacon’s movies in KBMovies Data Set*

*#Get Bacon’s CO Stars*

*hpcc.filter(output=0,data='ActorsInMovies',out.dataframe='CoStars',condition='Movie IN SET(KBMovies,Movie)')*

*hpcc.filter(output=0,data='CoStars',condition="actor<>'Kevin Bacon'",out.dataframe='CoStars2')*

*#Now delete all the duplicate records*

*hpcc.dedup(output=0,data='CoStars2',out.dataframe='KBCoStars',condition='actor',all='ALL')*

*#* *Get KBacon Costars' Movies*

*#* *First find all of the movies that a KBCoStar has been in CSM*

*hpcc.join(output=0,out.dataframe='CSM2',Dataset1='ActorsInMovies',Dataset2='KBCoStars',joinCondition='LEFT.actor=RIGHT.actor',fields='ActorsInMovies',type='LOOKUP')*

*hpcc.dedup(output=0,dataframe='CSM2',out.dataframe='CSM',condition='movie',all='ALL')*

*# Now we need to remove all of those that KB was in himself. We can use a set; KB has not been in (quite!) that many movies*

*hpcc.filter(output=0,data='CSM',out.dataframe='KBCoStarMovies',condition='movie NOT IN SET(KBMovies,movie)')*

*#------ Bacon # 2 Actors*

*# To be a Co2Star of Kevin Bacon you must have appeared in a movie with a CoStar of Kevin Bacon*

*# This corresponds to having a Bacon number of 2*

*hpcc.join(output=0,out.dataframe='KBCo2S2',Dataset1='ActorsInMovies',Dataset2='KBCoStarMovies',joinCondition='LEFT.movie=RIGHT.movie',fields='ActorsInMovies',type='LOOKUP')*

*hpcc.dedup(output=0,dataframe='KBCo2S2',condition='actor',all='ALL',out.dataframe='KBCo2S')*

*# KCCo2S = ALL Actors appearing in Movies of KBacon's CoActors*

*# The above is all the people in the movies; but some will have been co-stars of KB*

*#directly - these must be removed*

*# The LEFT ONLY join removes items in one list from another*

*hpcc.join(output=0,out.dataframe='KBCo2Stars',Dataset1='KBCo2S',Dataset2='KBCoStars',joinCondition='LEFT.actor=RIGHT.actor',fields='KBCo2S',type='LEFT ONLY')*

*#------- bacon # 2 Movies*

*# Co2SM = what movies have all the Co2Stars been in?*

*hpcc.join(output=0,out.dataframe='Co2SM2',Dataset1='ActorsInMovies',Dataset2='KBCo2Stars',joinCondition='LEFT.actor=RIGHT.actor',fields='ActorsInMovies',type='LOOKUP')*

*hpcc.dedup(output=0,out.dataframe='Co2SM',dataframe='Co2SM2',condition='movie',all='ALL')*

*# Co2SM = ALL Movies KBCo2Stars have been in*

*# Of course some of these movies will have CoStars in too and thus will already have*

*#been listed. Note this list will not contain any Kevin Bacon movies OR the movie would*

*#have been reached earlier!*

*hpcc.join(output=0,out.dataframe='KBCo2StarMovies',Dataset1='Co2SM',Dataset2='KBCoStarMovies',joinCondition='Co2SM.movie=KBCoStarMovies.movie',fields='LEFT',type='LEFT ONLY')*

*#------ bacon #3 Actors*

*# Find people with a Bacon number of 3*

*# This code is very similar to KBCo2Stars; one might be tempted to common up into a*

*# function or macro.*

*hpcc.join(output=0,out.dataframe='KBCo3S2',Dataset1='ActorsInMovies',Dataset2='KBCo2StarMovies',joinCondition='ActorsInMovies.movie=KBCo2StarMovies.movie',fields='ActorsInMovies',type='LOOKUP')*

*hpcc.dedup(output=0,out.dataframe='KBCo3S',dataframe='KBCo3S2',condition='actor',all='ALL')*

*# KBCo3S = ALL CoStars in KBCo2Star Movies*

*# The above is all the people in the movies; but some will have been co2stars of KB*

*# directly - these must be removed. The LEFT ONLY join removes items in one list from*

*# another. There should not be any direct CoStars in this list (or the movie would have*

*# been a CoStarMovie not a CoCoStarMovie)*

*hpcc.join(output=0,out.dataframe='KBCo3Stars',Dataset1='KBCo3S',Dataset2='KBCo2Stars',joinCondition='LEFT.actor=RIGHT.actor',fields='KBCo3S',type='LEFT ONLY')*

*#----- bacon #3 Movies*

*# So what movies have all the KBCo3Stars been in?*

*hpcc.join(output=0,out.dataframe='Co3SM2',Dataset1='ActorsInMovies',Dataset2='KBCo3Stars',joinCondition='ActorsInMovies.actor=KBCo3Stars.actor',fields='ActorsInMovies',type='LOOKUP')*

*hpcc.dedup(output=0,out.dataframe='Co3SM',dataframe='Co3SM2',condition='movie',all='ALL')*

*# Co3SM = ALL Movies KBCo3Stars have been in*

*# Of course some of these movies will have KBCo2Stars in too and thus will already have*

*# been listed. Note We ONLY have to remove one level back from the list; previous levels*

*# cannot be reached by definition*

*hpcc.join(output=0,out.dataframe='KBCo3StarMovies',Dataset1='Co3SM',Dataset2='KBCo2StarMovies',joinCondition='Co3SM.movie=KBCo2StarMovies.movie',fields='Co3SM',type='LEFT ONLY')*

*#------bacon No 4 Actors*

*hpcc.join(output=0,out.dataframe='KBCo4S2',Dataset1='ActorsInMovies',Dataset2='KBCo3StarMovies',joinCondition='ActorsInMovies.movie=KBCo4StarMovies.movie',fields='ActorsInMovies',type='LOOKUP')*

*hpcc.dedup(output=0,out.dataframe='KBCo4S',dataframe='KBCo4S2',condition='actor',all='ALL')*

*hpcc.join(output=0,out.dataframe='KBCo4Stars',Dataset1='KBCo4S',Dataset2='KBCo3Stars',joinCondition='LEFT.actor=RIGHT.actor',fields='KBCo4S',type='LEFT ONLY')*

*#----- bacon #4 Movies*

*# So what movies have all the Co4Stars been in?*

*hpcc.join(output=0,out.dataframe='Co4SM2',Dataset1='ActorsInMovies',Dataset2='KBCo4Stars',joinCondition='LEFT.actor=RIGHT.actor',fields='LEFT',type='LOOKUP')*

*hpcc.dedup(output=0,out.dataframe='Co4SM',dataframe='Co4SM2',condition='movie',all='ALL')*

*# Co4SM = ALL Movies KBCo4Stars have been in*

*# Of course some of these movies will have Co3Stars in too and thus will already have*

*# been listed. Note We ONLY have to remove one level back from the list; previous levels*

*# cannot be reached by definition*

*hpcc.join(output=0,out.dataframe='KBCo4StarMovies',Dataset1='Co4SM',Dataset2='KBCo3StarMovies',joinCondition='Co4SM.movie=KBCo3StarMovies.movie',fields='LEFT',type='LEFT ONLY')*

*#----- bacon #5 Stars*

*hpcc.join(output=0,out.dataframe='KBCo5S2',Dataset1='ActorsInMovies',Dataset2='KBCo4StarMovies',joinCondition='LEFT.movie=RIGHT.movie',fields='LEFT',type='LOOKUP')*

*hpcc.dedup(output=0,out.dataframe='KBCo5S',dataframe='KBCo5S2',condition='actor',all='ALL')*

*hpcc.join(output=0,out.dataframe='KBCo5Stars',Dataset1='KBCo5S',Dataset2='KBCo4Stars',joinCondition='LEFT.actor=RIGHT.actor',fields='LEFT',type='LEFT ONLY')*

*#----- bacon #5 Movies*

*hpcc.join(output=0,out.dataframe='Co5SM2',Dataset1='ActorsInMovies',Dataset2='KBCo5Stars',joinCondition='LEFT.actor=RIGHT.actor',fields='LEFT',type='LOOKUP')*

*hpcc.dedup(output=0,out.dataframe='Co5SM',dataframe='Co5SM2',condition='movie',all='ALL')*

*hpcc.join(output=0,out.dataframe='KBCo5StarMovies',Dataset1='Co5SM',Dataset2='KBCo4StarMovies',joinCondition='LEFT.movie=RIGHT.movie',fields='LEFT',type='LEFT ONLY')*

*#----- bacon #6 Stars*

*# Find people with a Bacon number of 6*

*# KBCo5 is getting small again - can move back down to the SET?*

*hpcc.filter(output=0,out.dataframe='KBCo6S2',data='ActorsInMovies',condition='movie IN SET(KBCo5StarMovies, movie)')*

*hpcc.dedup(output=0,out.dataframe='KBCo6S',dataframe='KBCo6S2',condition='actor',all='ALL')*

*hpcc.join(output=0,out.dataframe='KBCo6Stars',Dataset1='KBCo6S',Dataset2='KBCo5Stars',joinCondition='KBCo6S.actor=KBCo5Stars.actor',fields='LEFT',type='LEFT ONLY')*

*#----- bacon No 6 Movies*

*hpcc.filter(output=0,out.dataframe='Co6SM2',data='ActorsInMovies',condition='actor IN SET(KBCo6Stars, actor)',)*

*hpcc.dedup(output=0,out.dataframe='Co6SM',dataframe='Co6SM2',condition='movie',all='ALL')*

*hpcc.filter(output=0,out.dataframe='KBCo6StarMovies',data='Co6SM',condition='movie NOT IN SET(KBCo5StarMovies, movie)')*

*#----- bacon #7 Movies*

*hpcc.filter(output=0,out.dataframe='KBCo7S2',data='ActorsInMovies',condition='movie IN SET(KBCo6StarMovies,movie)')*

*hpcc.dedup(output=0,out.dataframe='KBCo7S',dataframe='KBCo7S2',condition='actor',all='ALL')*

*hpcc.filter(out.dataframe='KBCo7Stars',data='KBCo7S',condition='actor NOT IN SET(KBCo6Stars, actor)')*

*# For counting the number of variables in each of the data sets*

*hpcc.count('KBMoviesCount',in.data='KBMovies')*

*hpcc.count('KBCoStarsCount',in.data='KBCoStars')*

*hpcc.count('KBCoStarMoviesCount',in.data='KBCoStarMovies')*

*hpcc.count('KBCo2StarsCount',in.data='KBCo2Stars')*

*hpcc.count('KBCo2StarMoviesCount',in.data='KBCo2StarMovies')*

*hpcc.count('KBCo3StarsCount',in.data='KBCo3Stars')*

*hpcc.count('KBCo3StarMoviesCount',in.data='KBCo3StarMovies')*

*hpcc.count('KBCo4StarsCount',in.data='KBCo4Stars')*

*hpcc.count('KBCo4StarMoviesCount',in.data='KBCo4StarMovies')*

*hpcc.count('KBCo5StarsCount',in.data='KBCo5Stars')*

*hpcc.count('KBCo5StarMoviesCount',in.data='KBCo5StarMovies')*

*hpcc.count('KBCo6StarsCount',in.data='KBCo6Stars')*

*hpcc.count('KBCo7StarsCount',in.data='KBCo7Stars')*

*hpcc.count('KBCo6StarMoviesCount',in.data='KBCo6StarMovies')*