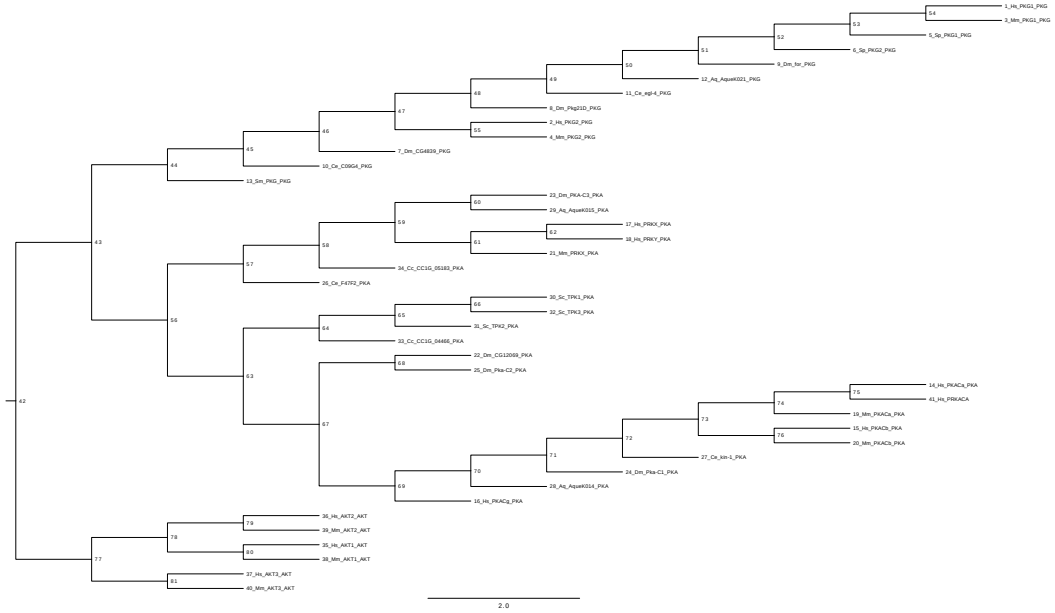
Kinase_divergence_toy_example

David Bradley

15 March 2019

Identification of functionally divergent residues in the PKA family

To illustrate the methods given in the manuscript, we present here all of the code for a 'toy' example wherein a simplified phylogeny of AKT, PKG, and PKA kinases has been used to identify all residues in the PKA kinase domain that are responsible for its divergence from PKG. The kinase phylogeny is given below (zoom-in to see the node labels):



The first step is to load in the MSA that was used to generate the phylogeny and the ancestral sequence reconstructions, along with a phylogeny with ancestral nodes labelled. The 'rst' file containing amino acid posterior probabilities for every site and ancestral node must also be included:

```
# Read in the kinase alignment file

fas <- read.fasta('PKA_PKG_AKT_named_PRKACA_al_trim_filter.fa')

# Read in the tree with ancestral nodes annotated by codeml

tree <- read.tree('kinase_toy_codeml_annotated.tre')

tips <- tree$tip.label
nodes <- tree$node.label

# Read in the 'rst' file containg the ancestral posterior probabilities
x <- readLines('rst_toy')
```

The next step is to identify all tips in the tree corresponding to the kinase family of interest (i.e. PKA):

```
# Specify the family of interest

family <- 'PKA'

# Retrieve the tip labels containing the family of interest
Group <- grep(family,tips)

# Families containing fewer than 5 sequences in this dataset are not considered further
if (length(Group) < 5) {next}

# Retrieve the sequences containing the family of interest
seqs <- (fas[[2]][grep(family,fas$id),])

# Sequence sample
seqs[1:3,1:50]
```

```
##              [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
## Hs_PKACa_PKA "F"  "E"  "R"  "I"  "K"  "T"  "L"  "G"  "T"  "G"   "S"
## Hs_PKACb_PKA "F"  "E"  "R"  "K"  "K"  "T"  "L"  "G"  "T"  "G"   "S"
## Hs_PKACg_PKA "F"  "E"  "R"  "L"  "R"  "T"  "L"  "G"  "M"  "G"   "S"
##              [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21]
## Hs_PKACa_PKA "F"   "G"   "R"   "V"   "M"   "L"   "V"   "K"   "H"   "K"
## Hs_PKACb_PKA "F"   "G"   "R"   "V"   "M"   "L"   "V"   "K"   "H"   "K"
## Hs_PKACg_PKA "F"   "G"   "R"   "V"   "M"   "L"   "V"   "R"   "H"   "Q"
##              [,22] [,23] [,24] [,25] [,26] [,27] [,28] [,29] [,30] [,31]
## Hs_PKACa_PKA "E"   "T"   "G"   "N"   "H"   "Y"   "A"   "M"   "K"   "I"
## Hs_PKACb_PKA "A"   "T"   "E"   "Q"   "Y"   "Y"   "A"   "M"   "K"   "I"
## Hs_PKACg_PKA "E"   "T"   "G"   "G"   "H"   "Y"   "A"   "M"   "K"   "I"
##              [,32] [,33] [,34] [,35] [,36] [,37] [,38] [,39] [,40] [,41]
## Hs_PKACa_PKA "L"   "D"   "K"   "Q"   "K"   "V"   "V"   "K"   "L"   "K"
## Hs_PKACb_PKA "L"   "D"   "K"   "Q"   "K"   "V"   "V"   "K"   "L"   "K"
## Hs_PKACg_PKA "L"   "N"   "K"   "Q"   "K"   "V"   "V"   "K"   "M"   "K"
##              [,42] [,43] [,44] [,45] [,46] [,47] [,48] [,49] [,50]
## Hs_PKACa_PKA "Q"   "I"   "E"   "H"   "T"   "L"   "N"   "E"   "K"
## Hs_PKACb_PKA "Q"   "I"   "E"   "H"   "T"   "L"   "N"   "E"   "K"
## Hs_PKACg_PKA "Q"   "V"   "E"   "H"   "I"   "L"   "N"   "E"   "K"
```

There are 22 PKA sequences overall in this dataset. A small sample of the PKA sequences is given directly above.

From the phylogeny tips, we should now identify the largest clade containing the kinase group of interest (i.e PKA):

```
# Separate into chunks

chunkvec <- 0
count <- 0

# Here I introduce a threshold that allows me to determine whether
# non-continuities in tip labels are due to the presence of 'contaminant'
# sequences within a clade or actually due to division of the family sequences
# into separate clades

thresh <- 10
```

```
# Here, the 'count' variable is used to track the clade membership of
#different sequences by recording discontinuities in the tip labels

for (i in 2:length(Group)) {

  if (Group[i]-Group[i-1] < thresh) {
    chunkvec <- c(chunkvec,count)
  }
  else {
    count = count+1
    chunkvec <- c(chunkvec,count)
  }
}

# Now we select from our group those sequences belonging to the largest clade

tab <- sort(table(chunkvec))
major_clade <- Group[which(chunkvec == names(tab)[length(tab)])]

# Clades containing fewer than 5 sequences in this dataset are not considered further
if(length(major_clade) < 5) {next}

print(length(major_clade))
```

```
## [1] 22
```

There are 22 kinases in this clade. We now need to make sure that all of them are PKAs:

```
# Determine the clade purity by calculating the 'frac' variable

globalpath <- nodepath(tree)

Query_anc <- getMRCA(tree,major_clade)
path <- nodepath(tree,from=getMRCA(tree,tips),to=Query_anc)
Query_anc_anc <- path[length(path)-1]

ancestral_path <- globalpath[grep(Query_anc,globalpath)]
frac <- length(intersect(major_clade,unlist(ancestral_path)))/length(ancestral_path)
major_clade_rev <- rev(major_clade)
frac_rev <-
length(intersect(major_clade_rev,unlist(ancestral_path)))/length(ancestral_path)
count=1

print(frac)
```

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

The clade is completely 'pure', meaning that it contains no AKT or PKG sequences. It is therefore not necessary to prune the clade.

We now need to find the ancestral node for the family of interest and its sister clade in the phylogeny (i.e. PKA and PKG, respectively):

```
# Now we need to find the ancestor node of the major clade

tips_new <- rapply(lapply(strsplit(tips[major_clade],split='_'),
```

```
          function(x) x[2:length(x)]), function(x) paste(x,collapse='_'))

if (rapply(strsplit(rownames(seqs), split=''), function(x) x[length(x)])[1] == '_') {
  tips_new <- paste(tips_new,'_',sep='')
}

# Retrieve all sequences from the major clade

seqs <- seqs[rownames(seqs) %in% tips_new,]

# Find the ancestral node of the major clade

Query_anc <- getMRCA(tree,major_clade)
path <- nodepath(tree,from=getMRCA(tree,tips),to=Query_anc)

# Find the node directly anetecedent to the ancestral node

Query_anc_anc <- path[length(path)-1]
globalpath <- nodepath(tree)
ancestral_path <- globalpath[grep(Query_anc_anc,globalpath)]

# Find the sister clade and its ancestral node

treepos <- grep(Query_anc_anc,ancestral_path[[1]])
Sister_anc <-
  setdiff(unique(rapply(ancestral_path, function(x) x[treepos+count])),Query_anc)
```

PKA ancestral node:

```
## [1] 56
```

PKG ancestral node:

```
## [1] 44
```

If we refer back to the phylogeny above (zoom-in), we can see that the code has correctly retrieved the PKA and PKG ancestral nodes.

We now use the following code to extract amino acid posterior probabilities for every site for the two nodes in question:

Posterior probabilities for the first MSA site is given in both cases for a sample of amino acids:

```
rapply(strsplit(PPs_q[[1]],split=' '), function(x) x[c(15,3,5,7,9,11,13)])
```

```
## [1] "F(0.997)" "R(0.000)" "D(0.000)" "Q(0.000)" "G(0.000)" "I(0.000)"
## [7] "K(0.000)"
```

```
rapply(strsplit(PPs_s[[1]],split=' '), function(x) x[c(15,3,5,7,9,11,13)])
```

```
## [1] "F(0.553)" "R(0.000)" "D(0.000)" "Q(0.000)" "G(0.000)" "I(0.022)"
## [7] "K(0.000)"
```

Next we must iterate through all of the MSA positions and calculate a divergence score:

```
# Now we must iterate through every one of the kinase domain positions
# and calculate a divergence score between the query clade
# and the sister clade.
```

```
scorevec <- NULL

aavec <- c('A', 'R', 'N', 'D', 'C', 'Q', 'E', 'G', 'H', 'I',
           'L', 'K', 'M', 'F', 'P', 'S', 'T', 'W', 'Y', 'V')

for (i in 1:length(PPs_query)) {

  # Most likely query aa (ancestral) and associated probability
  query_aa <- aavec[which(PPs_query[[i]] == max(PPs_query[[i]]))][1]
  query_aa_prob <- max(PPs_query[[i]])

  # Calculate the residue conservation at this site also
  query_aa_cons <- conserv(seqs,method='similarity')[i]
  query_product <- query_aa_cons

  # Most likely sister aa (ancestral) and associated probability
  sister_aa <- aavec[which(PPs_sister[[i]] == max(PPs_sister[[i]]))][1]
  sister_aa_prob <- NULL

  # The divergence score is calculated differently depending
  # on whether the predicted ancestral amino acid
  # for the sister clade is the same or different
  # from that of the query clade

  if (query_aa == sister_aa) {
    #Probability that ancestral amino acid
    #in the sister is the same as that for the query
    multiplier <- 1
    sister_aa_prob <- max(PPs_sister[[i]])
  } else {
    #Probability that ancestral amino acid in the sister is
    # different from that of the query
    multiplier <- -1
    sister_aa_prob <- sum(as.numeric(PPs_sister[[i]][which(aavec != query_aa)]))
  }

  sister_product <- multiplier*as.numeric(sister_aa_prob)

  score <- query_product - sister_product
  scorevec <- c(scorevec, score)
}

scorevec <- round(scorevec,3)
```

Let's have a look now at the five highest-scoring alignment positions:

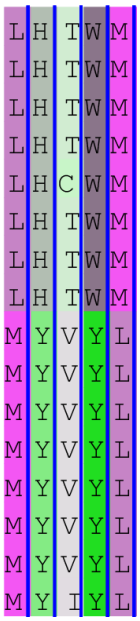
```
order(scorevec,decreasing=TRUE)[1:5]
```

```
## [1] 179  53 189 141 137
```

```
sort(scorevec,decreasing=TRUE)[1:5]
```

```
## [1] 1.886 1.776 1.715 1.626 1.598
```

The sequence sample of the relevant sites (in numerical order) is as follows (PKA top, PKG bottom):



The final stap is to map these residues to the protein kinase structure. This can be achieved by making use of the 'colnumbering.txt' file that takes into account residues that were trimmed from the alignment:

```
detach("package:bio3d", unload=TRUE)
library(seqinr)
```

```
##
## Attaching package: 'seqinr'
```

```
## The following object is masked from 'package:Biostrings':
##
##     translate
```

```
## The following objects are masked from 'package:ape':
##
##     as.alignment, consensus
```

```
noneg <- scorevec

fasta_al <- read.fasta('PKA_PKG_AKT_named_PRKACA_al.fa')

count = 0
almap <- numeric(3)

# Extract the human PRKACA sequence
PDB_al <-fasta_al[[length(fasta_al)]][1:length(fasta_al[[length(fasta_al)]])]

# Iterate through each alignment position,
# and record whenever there is a 'gap' in the human PRKACA sequence

for (n in 1:length(PDB_al)) {

  if (PDB_al[n] == "-") {

    count = count
    pair_vec <- c(n,'-','')
```

```
    almap <- rbind(almap,pair_vec)
  }

  else {

    count = count + 1
    pair_vec <- c(n,count,'')
    almap <- rbind(almap,pair_vec)
  }
}
almap <- almap[-1,]

# For mapping, we only need to retain positions
# that were kept in the trimmed alignment after trimAl processing

colmap <- read.table('colnumbering.txt',col.names=FALSE,stringsAsFactors = FALSE,sep=',')
colmap <- as.numeric(colmap[,1])
colmap <- colmap+1
trim_map <- almap[as.numeric(almap[,1]) %in% colmap,]

# Now the aggregated scores obtained previously
# can be mapped to the human PRKACA sequence
names(noneg) <- trim_map[,2]
```

Now extract the human UniProt PRKACA numbers corresponding to the five highest-scoring sites:

```
noneg[order(scorevec,decreasing=TRUE)[1:5]]
```

```
##   222    96   232   184   180
## 1.886 1.776 1.715 1.626 1.598
```

Now subtract -1 to map the sites to PDB 1atp:

```
as.numeric(names(noneg[order(scorevec,decreasing=TRUE)[1:5]]))-1
```

```
## [1] 221  95 231 183 179
```

Now map to PDB:1atp using pyMOL: