Simulated tree - Feb 2016

S. Le Vu

(Dated: February 23, 2016)

```
library(ape)
library(ggplot2)
```

I. LOAD STUFF

Import simulation outputs

```
## Load newest Rdata
l <- list.files(pattern="*.Rdata") # list.files(pattern="Rdata$") list.files(pattern="out")
load(l[length(l)])
ls()

[1] "l"     "o"     "parms" "tree"

tree


Phylogenetic tree with 12164 tips and 10671 internal nodes.

Tip labels:
7, 8, 9, 10, 11, 12, ...

Unrooted; includes branch lengths.
```

Import ExaML tree 0

```
t_uk <- read.tree(file = "../phylo-uk/data/ExaML_result.subUKogC_noDRM.finaltree.000")
## drop OG
og <- c("Ref1", "Ref2", "Ref3", "Ref4", "Ref5", "Ref6", "HXB2")
t_uk <- drop.tip(t_uk, og )
t_uk


Phylogenetic tree with 12164 tips and 12163 internal nodes.

Tip labels:
34695, 21677, 72292, 81292, 85197, 53538, ...

Rooted; includes branch lengths.
```

II. TREE DISTANCES

Patristic distances ?

```
####  cluster size to real data. Need to have same number of clusters ?

## get distances
```

```
##- matrix first into distances

#- sim tree
if (file.exists("data/simtree_dist.rds")){
  dsimtree <- readRDS("data/simtree_dist.rds")
} else {
dsimtree <- as.dist(cophenetic.phylo(tree))
saveRDS(dsimtree, file = "data/simtree_dist.rds")
}
# uk tree
if (file.exists("data/uktree_dist.rds")){
  duktree <- readRDS("data/uktree_dist.rds")
} else {
  duktree <- as.dist(cophenetic.phylo(t_uk))
  saveRDS(duktree, file = "data/uktree_dist.rds")
}
head(dsimtree)
```

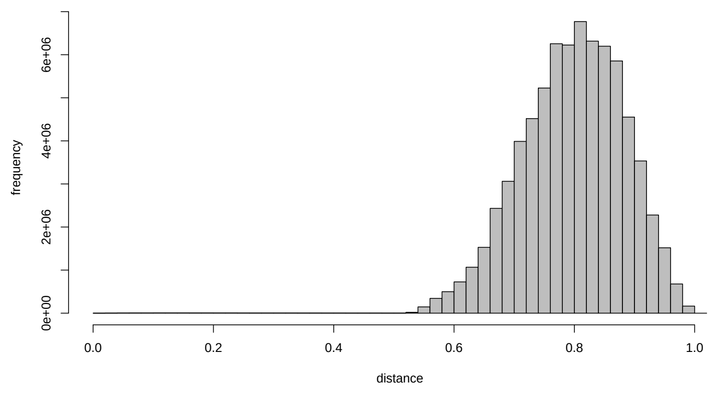
```
[1] 24930 24930 24930 24930 24930 24930
```

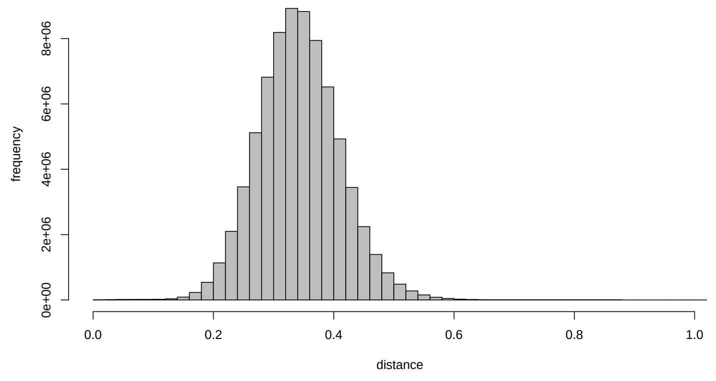
```
head(duktree)
```

```
[1] 2.000001e-06 8.505254e-02 7.145780e-02 1.257209e-01 1.010726e-01 1.104545e-01
```

```
## normalize
simx <- dsimtree / (max(dsimtree) - min(dsimtree))
ukx <- duktree / (max(duktree) - min(duktree))
# rm(simtree, uktree)

##- histogram distances
# summary(x)
hist(simx, breaks = 50, xlab = "distance", ylab = "frequency", main = "", col = "grey")
hist(ukx, breaks = 50, xlab = "distance", ylab = "frequency", main = "", col = "grey")
```





III. CLUSTERING

Clustering with fixed number of groups ?

```
simhc <- hclust(simx, method = "average") # UPGMA
ukhc <- hclust(ukx, method = "average")

##- cut based on height
if (F){
#- function of heights
nheights <- 10 # number of threshold
up <- round(mean(simx), 1) # cut up to the mean
# breaks <-  seq(1/nbreaks, 1-(1/nbreaks), by = 1/nbreaks)
simclus <- cutree(simhc, h = seq(up / nheights, up, by = up / nheights) ) # h = breaks
up <- round(mean(ukx),1) # cut up to the mean
ukclus <- cutree(ukhc,  h = seq(up / nheights, up, by = up /nheights) )
# rm(simx, ukx)
}

#- cut as function of k groups
kgroups <- c(500, 1000, 5000, 8000, 10000)
simclus <- cutree(simhc, k = kgroups ) # h = breaks
ukclus <- cutree(ukhc,  k = kgroups )
# colnames(simclus) <- paste("k",colnames(simclus),sep='')
# colnames(ukclus) <- paste("k",colnames(ukclus),sep='')
head(simclus)
```

```
   500 1000 5000 8000 10000
7    1    1    1    1     1
8    2    2    2    2     2
9    3    3    3    3     3
10   4    4    4    4     4
11   5    5    5    5     5
12   6    6    6    6     6

head(ukclus)

      500 1000 5000 8000 10000
34695   1    1    1    1     1
21677   1    1    1    1     1
72292   1    1    2    2     2
81292   1    1    1    3     3
85197   1    1    3    4     4
53538   1    1    4    5     5

##- Calculate size(=Freq) of each cluster across different threshold
simfreqClust <- apply(simclus, 2, function(x) as.data.frame(table(x))) # list
ukfreqClust <- apply(ukclus, 2, function(x) as.data.frame(table(x)))
# str(simfreqClust)
# head(simfreqClust[[1]])

##- number of different clusters by threshold # if number varies !
# sapply(simfreqClust, function(x) dim(x)[1])
# sapply(ukfreqClust, function(x) dim(x)[1])

##- cluster size
#- sim
sapply(simfreqClust, function(x) summary(x$Freq))

            500    1000   5000  8000 10000
Min.       1.00    1.00  1.000  1.00 1.000
1st Qu.    2.00    2.00  1.000  1.00 1.000
Median     6.00    6.00  2.000  1.00 1.000
Mean      24.33   12.16  2.433  1.52 1.216
3rd Qu.   13.00   13.00  3.000  2.00 1.000
Max.    7395.00 3019.00 20.000 11.00 7.000

#- uk
sapply(ukfreqClust, function(x) summary(x$Freq))

            500    1000    5000  8000  10000
Min.       1.00    1.00   1.000  1.00  1.000
1st Qu.    1.00    1.00   1.000  1.00  1.000
Median     2.00    2.00   1.000  1.00  1.000
Mean      24.33   12.16   2.433  1.52  1.216
3rd Qu.    7.00    6.00   2.000  1.00  1.000
Max.    6868.00 3486.00 142.000 86.00 39.000

##- percentiles
# sapply(freqClust, function(x) round(quantile(x$Freq,
# probs = c(0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 0.95, 0.99, 1))))
#
```

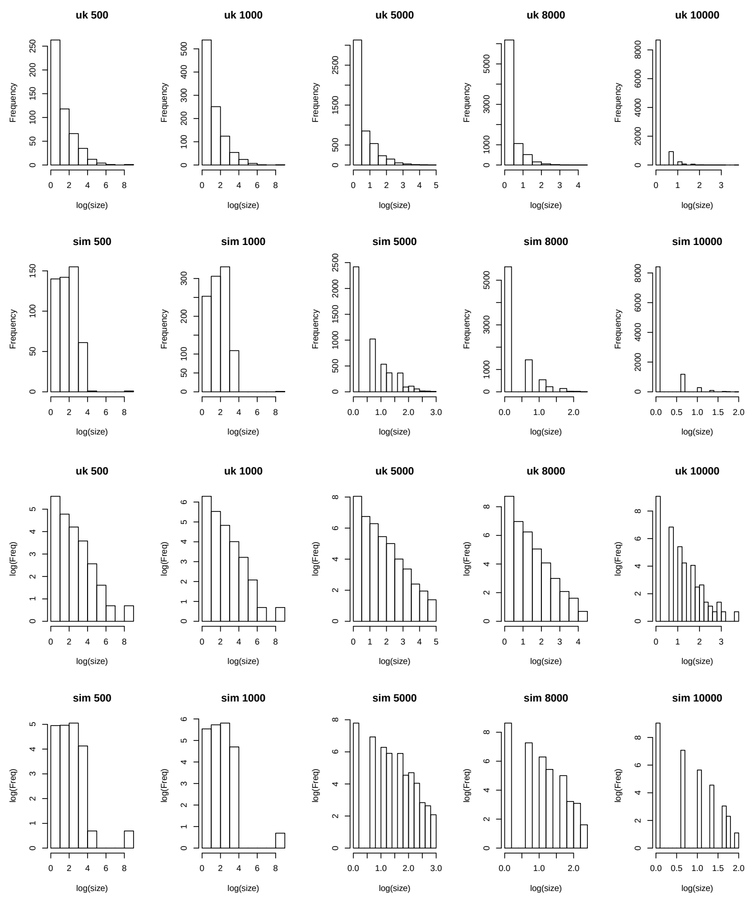
Distribution of sizes - semitransformed, transformed

```
##- distr of cluster sizes: log(x) and Y untransformed
par(mfcol=c(2, length(kgroups)))
for (i in 1:length(kgroups)){
  h <- hist(log(ukfreqClust[[i]]$Freq),
       main = paste("uk", names(ukfreqClust)[i]),
       xlab = "log(size)")
  hist(log(simfreqClust[[i]]$Freq),
       main = paste("sim", names(simfreqClust)[i]),
       xlab = "log(size)")
}

##- distr of cluster sizes: log(x) and log(y)
par(mfcol=c(2, length(kgroups)))
for (i in 1:length(kgroups)){
  h <- hist(log(ukfreqClust[[i]]$Freq), plot = F)
  h$counts <- log1p(h$counts) # log(y)
  plot(h, ylab = "log(Freq)",
       main = paste("uk", names(ukfreqClust)[i]),
       xlab = "log(size)")

  h <- hist(log(simfreqClust[[i]]$Freq), plot = F)
  h$counts <- log1p(h$counts) # log(y)
  plot(h, ylab = "log(Freq)",
          main = paste("sim", names(simfreqClust)[i]),
          xlab = "log(size)")

}
```

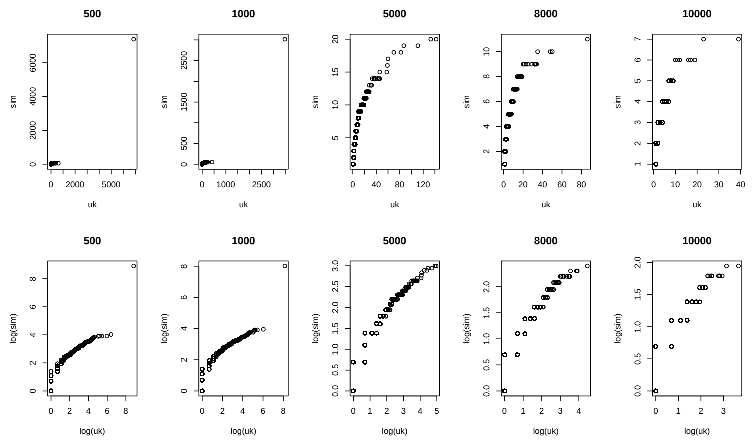


QQ plot

```
par(mfcol=c(2, length(kgroups)))
for (i in 1:length(kgroups)){
  qqplot(ukfreqClust[[i]]$Freq,
         simfreqClust[[i]]$Freq,
         main = names(ukfreqClust)[i],
         xlab = "uk", ylab = "sim")

  qqplot(log(ukfreqClust[[i]]$Freq),
         log(simfreqClust[[i]]$Freq),
         main = names(ukfreqClust)[i],
         xlab = "log(uk)", ylab = "log(sim)")

}
```



IV. ADD PATIENTS DATA

Add data from sample states and time

```
##- converting sample states in table of co-variates ?
demes <- as.vector(read.csv(file = "demes.csv")$x)
sampleTimes <- scan( file = 'sampleTimes' )
ss  <- matrix( scan( file = 'sampleStates' ) ,
               byrow = TRUE,
               ncol = length(demes))
colnames(ss) <- demes
dim(ss)

[1] 12164   121

max(ss[,121]) # nothing on source

[1] 0

demo <- data.frame()
for (i in 1:dim(ss)[1]){ # dim(ss)[1]
  deme <- names(which(ss[i,] == 1)) # name of column which has value 1
  patient <- i
  time <- sampleTimes[i]
  age <- as.numeric( regmatches( deme,
                regexec( "\\.age([0-9])", deme) )[[1]][2] )
  care <- as.numeric( regmatches( deme,
                regexec( "care([0-9])", deme) )[[1]][2] )
  stage <- as.numeric( regmatches( deme,
                regexec( "stage([0-9])", deme) )[[1]][2] )
  risk <- as.numeric( regmatches( deme,
                regexec( "riskLevel([0-9])", deme) )[[1]][2] )
  demo <- rbind(demo, cbind(
    patient, time, age, care, stage, risk))
}
str(demo)

'data.frame': 12164 obs. of  6 variables:
 $ patient: num  1 2 3 4 5 6 7 8 9 10 ...
 $ time   : num  11245 8627 6679 8446 8900 ...
 $ age    : num  3 3 4 4 4 3 4 3 3 4 ...
 $ care   : num  1 1 1 1 1 1 1 1 1 1 ...
 $ stage  : num  1 1 1 5 4 2 5 2 3 5 ...
 $ risk   : num  1 1 1 1 1 1 1 1 2 1 ...

##- date of diagnosis ?
date0 <- as.Date('1979-01-01')
demo$datediag <- date0 + demo$time
min(demo$datediag)

[1] "1996-11-15"

max(demo$datediag)

[1] "2013-02-15"
```

Add cluster size by patient

```
##- function to calculate both numclus and sizeclus for each seqindex into a LIST
##- with same variable names
  ##- in list
  l <- list()
  for (i in 1:length(kgroups)) {

  #- cluster number
  numclus <- as.data.frame(simclus[, i])
  numclus <- cbind(rownames(numclus), numclus)
  colnames(numclus) <- c("id", "num")
  row.names(numclus) <- NULL
  # head(numclus)

  #- size of cluster
  a <- merge(x = numclus, y = simfreqClust[[i]],
             by.x = "num", by.y = "x",
             all.x = TRUE, sort = FALSE)
  #- binary clustering variable
  a$Clus <- ifelse(a$Freq > 1, 1, 0)
  #- colnames
  colnames(a)[which(colnames(a) =="Freq")] <- "size"
  colnames(a)[which(colnames(a) =="Clus")] <- "clus"
  l[[i]] <- a
  names(l)[i] <- names(simfreqClust[i])
  }

  rm(a, numclus)
```

```
# str(l)

##-proportion in or out clusters
sapply(l, function(x) round(prop.table(table(x$clus)),2))

   500 1000 5000 8000 10000
0 0.01 0.01  0.2 0.46  0.69
1 0.99 0.99  0.8 0.54  0.31

##- cluster sizes
sapply(l, function(x) summary(x$size))

         500   1000   5000   8000 10000
Min.       1    1.0  1.000  1.000 1.000
1st Qu.   22   11.0  2.000  1.000 1.000
Median  7395   21.0  4.000  2.000 1.000
Mean    4504  762.7  4.425  2.223 1.488
3rd Qu. 7395   52.0  6.000  3.000 2.000
Max.    7395 3019.0 20.000 11.000 7.000
```

```
listclus <- lapply(l, function(x)
merge(x, demo,
      by.x = "id", by.y = "patient",
      all.x = T, sort = FALSE))

# head(listclus[[4]])
# table(listclus[[4]]$clus)
```

V. REGRESSIONS

A. Logistic

```
##- model: clus ~ age +  stage + time + risk
##- care = 1 for all at diagnosis
## ex.
logit_model = "clus ~ age + stage + time + risk"
logit_model_std = "clus ~ scale(age) + scale(stage) + scale(time) + scale(risk)"
?glm
lapply(listclus, function(x) summary(glm(formula = logit_model_std,
                                 data = x,
                                 family = binomial(link = "logit"))))

$`500`

Call:
glm(formula = logit_model_std, family = binomial(link = "logit"),
    data = x)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.2932   0.1035   0.1089   0.1179   0.1598

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
```

```
(Intercept)   5.06851    0.11629  43.585   <2e-16 ***
scale(age)    0.07370    0.11120   0.663   0.5075    
scale(stage)  0.01809    0.11472   0.158   0.8747    
scale(time)  -0.09012    0.11632  -0.775   0.4385    
scale(risk)  -0.18421    0.10107  -1.823   0.0684 .  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 943.23  on 12163  degrees of freedom
Residual deviance: 939.00  on 12159  degrees of freedom
AIC: 949

Number of Fisher Scoring iterations: 8


$`1000`

Call:
glm(formula = logit_model_std, family = binomial(link = "logit"), 
    data = x)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-3.2188   0.1310   0.1442   0.1589   0.2238  

Coefficients:
             Estimate Std. Error z value Pr(>|z|)    
(Intercept)   4.55488    0.09111  49.994   <2e-16 ***
scale(age)    0.05994    0.08537   0.702   0.4826    
scale(stage)  0.02410    0.08834   0.273   0.7850    
scale(time)  -0.22805    0.09289  -2.455   0.0141 *  
scale(risk)  -0.16634    0.07873  -2.113   0.0346 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1456.7  on 12163  degrees of freedom
Residual deviance: 1445.4  on 12159  degrees of freedom
AIC: 1455.4

Number of Fisher Scoring iterations: 7


$`5000`

Call:
glm(formula = logit_model_std, family = binomial(link = "logit"), 
    data = x)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-1.8353   0.6524   0.6639   0.6708   0.6828  

Coefficients:
```

```
              Estimate Std. Error z value Pr(>|z|)
(Intercept)   1.393642   0.022721  61.338   <2e-16 ***
scale(age)    0.001913   0.022860   0.084    0.933
scale(stage)  0.002938   0.022910   0.128    0.898
scale(time)  -0.024301   0.022878  -1.062    0.288
scale(risk)   0.013449   0.022864   0.588    0.556
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 12135  on 12163  degrees of freedom
Residual deviance: 12134  on 12159  degrees of freedom
AIC: 12144

Number of Fisher Scoring iterations: 4


$`8000`

Call:
glm(formula = logit_model_std, family = binomial(link = "logit"),
    data = x)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.265  -1.245   1.095   1.111   1.142

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)   0.1585314  0.0181926   8.714   <2e-16 ***
scale(age)   -0.0006118  0.0183335  -0.033    0.973
scale(stage)  0.0180774  0.0183458   0.985    0.324
scale(time)  -0.0027158  0.0182654  -0.149    0.882
scale(risk)  -0.0204569  0.0181770  -1.125    0.260
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 16787  on 12163  degrees of freedom
Residual deviance: 16784  on 12159  degrees of freedom
AIC: 16794

Number of Fisher Scoring iterations: 3


$`10000`

Call:
glm(formula = logit_model_std, family = binomial(link = "logit"),
    data = x)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.9038  -0.8656  -0.8471   1.5116   1.6187
```

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)    
(Intercept)  -0.80789    0.01964 -41.132   <2e-16 ***
scale(age)    0.02309    0.01985   1.163    0.245    
scale(stage)  0.03683    0.01981   1.859    0.063 .  
scale(time)   0.02224    0.01975   1.126    0.260    
scale(risk)  -0.01789    0.01974  -0.906    0.365    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 15031  on 12163  degrees of freedom
Residual deviance: 15024  on 12159  degrees of freedom
AIC: 15034

Number of Fisher Scoring iterations: 4

# logistic <- function(x, m = logit_model){
#   fit <- glm(m , data = x,
#                  family = binomial(link = "logit"))
#   co <- coef(summary(fit))
#   ## odds ratios and 95% CI
#   # or <- exp(cbind(OR = coef(fit), confint(fit)))
#   # return(list(co, or))
#   return(cbind(co[,c(1,4)]))
# }
#
# ##- test 1 level
# c <- listclus[[1]]
# logistic(x = c, m = logit_model_std)
#
# ##- all levels
# lapply(listclus, function(x) logistic(x, m = logit_model_std))
```

B. Linear

```
lm_model = "size ~ age + stage + time + risk"
lm_model_std = "scale(size) ~ scale(age) + scale(stage) + scale(time) + scale(risk)"

lapply(listclus, function(x) summary(lm(lm_model_std, data = x)))

$`500`

Call:
lm(formula = lm_model_std, data = x)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.2705 -1.2414  0.7943  0.8052  0.8286 

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)  -2.824e-14  9.068e-03   0.000    1.000
scale(age)   -4.245e-04  9.139e-03  -0.046    0.963
scale(stage) -6.856e-03  9.144e-03  -0.750    0.453
scale(time)   4.858e-03  9.104e-03   0.534    0.594
scale(risk)  -2.027e-03  9.070e-03  -0.224    0.823

Residual standard error: 1 on 12159 degrees of freedom
Multiple R-squared:  8.078e-05,Adjusted R-squared:  -0.0002482
F-statistic: 0.2456 on 4 and 12159 DF,  p-value: 0.9125


$`1000`

Call:
lm(formula = lm_model_std, data = x)

Residuals:
    Min      1Q  Median      3Q     Max
-0.6277 -0.5808 -0.5659 -0.5336  1.7610

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)  -2.208e-14  9.067e-03   0.000    1.000
scale(age)   -1.427e-02  9.138e-03  -1.562    0.118
scale(stage) -1.947e-03  9.144e-03  -0.213    0.831
scale(time)   1.686e-03  9.103e-03   0.185    0.853
scale(risk)   1.506e-03  9.070e-03   0.166    0.868

Residual standard error: 1 on 12159 degrees of freedom
Multiple R-squared:  0.0002211,Adjusted R-squared:  -0.0001078
F-statistic: 0.6724 on 4 and 12159 DF,  p-value: 0.6111


$`5000`

Call:
lm(formula = lm_model_std, data = x)

Residuals:
    Min      1Q  Median      3Q     Max
-1.0397 -0.6975 -0.1676  0.4660  4.5356

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.744e-16  9.067e-03   0.000   1.0000
scale(age)    8.351e-04  9.137e-03   0.091   0.9272
scale(stage) -2.079e-04  9.143e-03  -0.023   0.9819
scale(time)   3.411e-03  9.103e-03   0.375   0.7079
scale(risk)   1.969e-02  9.069e-03   2.171   0.0299 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1 on 12159 degrees of freedom
Multiple R-squared:  0.0004,Adjusted R-squared:  7.114e-05
F-statistic: 1.216 on 4 and 12159 DF,  p-value: 0.3015


$`8000`
```

```
Call:
lm(formula = lm_model_std, data = x)

Residuals:
    Min      1Q  Median      3Q     Max 
-0.7774 -0.7385 -0.1467  0.4663  5.3336 

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.257e-15  9.068e-03   0.000    1.000
scale(age)   -5.272e-03  9.138e-03  -0.577    0.564
scale(stage)  8.204e-03  9.144e-03   0.897    0.370
scale(time)   7.313e-03  9.104e-03   0.803    0.422
scale(risk)  -1.942e-03  9.070e-03  -0.214    0.830

Residual standard error: 1 on 12159 degrees of freedom
Multiple R-squared:  0.000139,Adjusted R-squared:  -0.0001899 
F-statistic: 0.4227 on 4 and 12159 DF,  p-value: 0.7924


$`10000`

Call:
lm(formula = lm_model_std, data = x)

Residuals:
    Min      1Q  Median      3Q     Max 
-0.6097 -0.5563 -0.5292  0.5587  6.2095 

Coefficients:
               Estimate Std. Error t value Pr(>|t|)  
(Intercept)  -1.373e-15  9.065e-03   0.000   1.0000  
scale(age)    9.973e-03  9.136e-03   1.092   0.2750  
scale(stage)  2.187e-02  9.142e-03   2.392   0.0168 *
scale(time)   9.734e-03  9.101e-03   1.070   0.2848  
scale(risk)   2.170e-03  9.067e-03   0.239   0.8109  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9998 on 12159 degrees of freedom
Multiple R-squared:  0.0006847,Adjusted R-squared:  0.000356 
F-statistic: 2.083 on 4 and 12159 DF,  p-value: 0.08024
```