**Note:** For further details on the full dataset of the network meta-analysis about epidermal growth factor receptor tyrosine kinase inhibitors-based therapies of EGFR-mutated non-small-cell lung cancer please contact the contact author (Professor Xiaohong Kang) at 1fy2014036@xxmu.edu.cn.

**Code for network meta-analysis**

library(netmeta)

pw <- pairwise(treat = alloc1, n = sampleSize, event = responders, studlab = study, data = data, sm = "OR")

netconnection(pw)

net <- netmeta(pw, ref = "Placebo", comb = FALSE)

netgraph(net, seq = "optimal",

plastic = FALSE,

number = FALSE,

offset = ifelse(n.trts < 1500, 0.025, 0.05),

labels = trts,

scale = 1.15,

multiarm = FALSE,

col = "black",

thickness = "number.of.studies",

lwd.min = 1, lwd.max = 10)

forest(net)

league <- netleague(net, digits = 2, seq = netrank(net, small.values = "bad"),bracket = "(", separator = " to ")

netsplit(net)

funnel(net)

**Code for rare event in NMA**

pw1 <- pairwise(treat = alloc, n = sampleSize, event = responders, studlab = study, data = data1, sm = "OR")

IV = netmetabin(pw, ref = "Placebo", method = "Inverse",details.chkmultiarm = TRUE)

netgraph(IV)

forest(IV)

league <- netleague(IV, digits = 2, seq = netrank(net, small.values = "bad"),bracket = "(", separator = " to ")

netsplit(IV)

funnel(IV)

**Code for pairwise meta-analysis**

library(meta)

csm <- metabin(event.e,n.e,event.c,n.c,data=data,sm="OR",studlab=Study,random=TRUE,robust=TRUE)

forest(csm, layout = "RevMan5", sortvar = csm$TE)

funnel(csm)

**Code for pooled incidence heatmap**

library(ComplexHeatmap)

library(circlize)

library(GetoptLong)

ht = Heatmap(data, col = col\_fun,

border\_gp = gpar(col = "black"), rect\_gp = gpar(col = "dimgray", lwd = 0.7),

show\_row\_names = FALSE, row\_names\_gp = gpar(fontsize = 10),

cluster\_columns = F, cluster\_rows = F, column\_names\_rot = 45,

column\_names\_gp = gpar(fontsize = 10), heatmap\_width = unit(8.5, "cm"), #6

heatmap\_height = unit(14, "cm"), show\_heatmap\_legend = FALSE,

row\_title = NULL,

left\_annotation = rowAnnotation(Incidence = anno\_boxplot(data, height = unit(2, "cm"), gp = gpar(fill = col1[1]))))

ht3 = ht + ht1

**Code for single-arm meta-analysis**

library(meta)

meta\_prop <- metaprop(event.e,n.e,data=data,studlab = paste(data$author,data$year,sep=","),sm="PRAW",incr=0.5,allincr=TRUE,addincr=FALSE, byvar = alloc)

forest(meta\_prop, layout = "RevMan5", col.square = "#0000ff", col.square.lines = "#000000", col.inside = "#000000", col.diamond = "#000000", col.diamond.lines = "#000000")

**Code for Bayesian network analysis**

library(gemtc)

library(rjags)

network <- mtc.network(data)

model <- mtc.model(network, type = "consistency", n.chain = 4, likelihood = "binom", link = "logit", linearModel = "random", dic = TRUE)

results <- mtc.run(model, n.adapt = 100000, n.iter = 200000, thin = 10)

forest(relative.effect(results, "Placebo"))

plot(results)

gelman.plot(results)

gelman.diag(results)

ranks <- rank.probability(results)

result.node <- mtc.nodesplit(network, thin=1)

summary.ns <- summary(result.node)

resultanohe <- mtc.anohe(network, n.adapt=5000, n.iter=20000, thin=1, n.chain=4, likelihood="binom", link="logit", linearModel="random")

relative.effect.table(results)

**Code for regression analysis**

library(gemtc)

**Binary variables**

networkreg <- mtc.network(data=data, studies=study)

model <- mtc.model(networkreg, type="regression”,

regressor=list(coefficient='unrelated',

variable='group',

control='Placebo'))

results <- mtc.run(model, n.adapt = 100000, n.iter = 200000, thin = 10)

print(summary(results))

print(gelman.diag(results))

ranks <- rank.probability(results, covariate = i, preferredDirection = 1)

test <- summary(results)

test2\_1$OR <- exp(test2\_1$Mean)

test2\_1$CI\_Lower <- exp(test2\_1$Mean - Z \* test2\_1$Time.series.SE)

test2\_1$CI\_Upper <- exp(test2\_1$Mean + Z \* test2\_1$Time.series.SE)

**Continuous variables**

networkreg <- mtc.network(data=data1, studies=study)

model <- mtc.model(networkreg, type="regression”,

regressor=list(coefficient='unrelated',

variable=colnames(data1)[5],

control='Placebo'))

results <- mtc.run(model, n.adapt = 100000, n.iter = 200000, thin = 10)

print(summary(results))

print(gelman.diag(results))

plotCovariateEffect(result)

**Codes for literature screening**

import os

import pandas

import re

disease\_type = r'PHASE.\*?TRIAL'

disease\_type1 = r'PHASE.\*?STUDY'

design = r'COMPARE.\*?PATIENTS'

people = r'PATIENTS.\*? CANCER'

people1 = r'WITH.\*? SCLC'

method = r'METHODS.\*?RESULTS'

method1 = r'METHODS.\*?FINDINGS'

efficency = r'FINDINGS.\*?ADVERSE EVENTS'

efficency1 = r'RESULTS.\*?ADVERSE EVENTS'

safety\_results = r'ADVERSE EVENT.\*? CONCLUSIONS'

safety\_results1 = r'ADVERSE EVENT.\*? INTERPRETATION'

safety\_results2 = r'ADVERSE EVENT.\*'

safety\_results3 = r'ADVERSE EVENTS WERE .\*?.'

test\_all1 = data

for i in test\_all1.index:

text = str(test\_all1.loc[i, 'Abstract Note'])

text = text.upper()

pattern1 = r'NCT\d+'

pattern2 = r'JRCTS\d+'

pattern3 = r'UMIN\d+'

pattern4 = r'ChiCTR\d+'

pattern5 = r'CTIS\d+'

pattern6 = r'EUCTR\d+'

pattern7 = r'JRCT\d+'

pattern8 = r'JPRN\d+'

pattern9 = r'JapicCTI\d+'

pattern10 = r'NL-OMON\d+'

pattern11 = r'PER-\d+'

#print(type(text))

if text is not None:

matches = re.findall(f"({pattern1}|{pattern2}|{pattern3}|{pattern4}|{pattern5}|{pattern6}|{pattern7}|{pattern8}|{pattern9}|{pattern10}|{pattern11})", str(text))

print(matches)

test\_all1.loc[i, 'Number'] = str(matches)

matches = re.findall(f"({disease\_type}|{disease\_type1})", str(text))

print(matches)

test\_all1.loc[i, 'Disease'] = str(matches)

matches = re.findall(design, str(text))

print(matches)

test\_all1.loc[i, 'Design'] = str(matches)

matches = re.findall(f"({people}|{people1})", str(text))

print(matches)

test\_all1.loc[i, 'Person'] = str(matches)

matches = re.findall(f"({method}|{method1})", str(text))

print(matches)

test\_all1.loc[i, 'Method'] = str(matches)

matches = re.findall(f"({efficency}|{efficency1})", str(text))

print(matches)

test\_all1.loc[i, 'Efficency'] = str(matches)

matches = re.findall(f"({safety\_results}|{safety\_results1}|{safety\_results2}|{safety\_results3})", str(text))

print(matches)

test\_all1.loc[i, 'SafetyResults'] = str(matches)

train\_data = test\_all1[test\_all1['Filter'].isin(['Clinical trials','Case report','Meta-analysis','Review','basic research','Real-World Study','book','observational','conferencePaper'])]

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import SVC

X = train\_data['Abstract Note'].fillna(' ')

y = train\_data['Filter']

vectorizer = TfidfVectorizer()

X\_vectorized = vectorizer.fit\_transform(X)

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X\_vectorized, y, test\_size=0.2, random\_state=42)

classifier = SVC()

classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_val)

accuracy = (y\_pred == y\_val).mean()

new\_data = test\_all1[~test\_all1['Filter'].isin(['Clinical trials','Case report','Meta-analysis','Review','basic research','Real-World Study','book','observational','conferencePaper'])]

X\_new = vectorizer.transform(new\_data['Abstract Note'].fillna(' '))

y\_new\_pred = classifier.predict(X\_new)

new\_data.loc[:, 'predicted\_Type'] = y\_new\_pred

print(new\_data.columns)