前期数据探索分析使用的 R 脚本：

在 CMTI 内部实验室服务器集群完全搭建成功并正常工作之前，我们尝试采用一些由 R 语言编写的程序对 SCCP 数据进行初步的探索，虽然当时的分析主要基于 DPC（Destination Point Code）和 OPC（Originating Point Code），但我们还是决定将这些 R 脚本记录在这里，包括：

1. 抽取 User error == Roaming not allowed 的行，然后把结果存入输出文件，因为出现这种用户错误的客户有很大可能是我们的潜在顾客，非常值得去考察隐藏用户的分布情况：

```
filenames <- list.files("sccp", pattern = "*.csv", full.names = TRUE)
library(plyr)
library(dostats)
outFile <- "numbers.csv"

# Keep only some useful fields

outFileColNames <- c("DPC", "OPC", "Cg", "Cd", "MSISDN", "Back", "Serv")
res <- NULL
for (i in seq_along(filenames)) {
  f <- read.csv(filenames[i])

# Retrieve the records with User error == Roaming not allowed

  res <- rbind(res, f[f[, 23] == "Roaming not allowed", c(7, 8, 41, 42, 43, 51, 67)])
}

# Save the result into the output file

write.table(res,file=outFile,sep=',',row.names=FALSE,col.names=outFileColNames,quote=FALSE)
```

2. 创建新的表格以将呼叫方的 Cg Sccp Add 与被叫方的 Cd Sccp Add 映射到相应的国家、地区或省市代码：

```
# Create a lookup table for the region code


dict_filename <- "sccp_dict.txt"
sccp_dict_tmp <- read.table(dict_filename, sep = "|", header = FALSE)
```

```
maxlen <- 0
sccp_dict <- new.env()
len <- dim(sccp_dict_tmp)[1]
headers <- NULL
headers <- rbind(headers, c("prefix", "area", "region", "carrier", "network"))
sccp_dict_out_file <- "sccp.dict.txt"
#write.table(as.data.frame(headers), file = sccp_dict_out_file, append = FALSE, sep = '|', row.names =
FALSE, col.names = FALSE, quote = FALSE)
for (i in seq(0, len / 8 - 1)) {
  codelist_str <- as.character(sccp_dict_tmp[i * 8 + 6, 1])
  codelist = strsplit(codelist_str, ",")
  area <- as.character(sccp_dict_tmp[i * 8 + 3,])
  for (j in 1:nchar(area)) {
    char <- substring(area, j, j)
    if (((char >= 'a') && (char <= 'z')) || ((char >= 'A') && (char <= 'Z'))) {
      break
    }
  }
  #print(area)
  region <- substring(area, j, nchar(area))
  #print(region)
  carrier <- as.character(sccp_dict_tmp[i * 8 + 4,])
  network <- as.character(sccp_dict_tmp[i * 8 + 5,])
  for (j in 1:length(codelist[[1]])) {
    if (nchar(codelist[[1]][j]) > maxlen) {
      maxlen = nchar(codelist[[1]][j])
    }
    sccp_dict[[codelist[[1]][j]]] = c(area, region, carrier, network)
    results <- NULL
    results <- rbind(results, c(codelist[[1]][j], area, region, carrier, network))
    #write.table(as.data.frame(results), file = sccp_dict_out_file, append = TRUE, sep = '|', row.names =
FALSE, col.names = FALSE, quote = FALSE, fileEncoding = "UTF-8")
  }
}
print ("maxlen")
print(maxlen)

# Create a lookup table for the province code

prov_filename <- "sccp_prov.txt"
sccp_prov_tmp <- read.table(prov_filename, sep = "|", header = FALSE)
sccp_prov <- new.env()
len_prov <- dim(sccp_prov_tmp)[1]
head_prov <- NULL
head_prov <- rbind(head_prov, c("code", "province"))
sccp_prov_out_file <- "sccp.prov.txt"
#write.table(as.data.frame(head_prov), file = sccp_prov_out_file, append = FALSE, sep = '|', row.names
= FALSE, col.names = FALSE, quote = FALSE)
```

```
for (k in seq(0, len_prov / 2 - 1)) {
  code <- as.character(sccp_prov_tmp[k * 2 + 2, 1])
  prov <- as.character(sccp_prov_tmp[k * 2 + 1, ])
  sccp_prov[[code]] <- prov
  result <- NULL
  result <- rbind(result, c(code, prov))
  #write.table(as.data.frame(result), file = sccp_prov_out_file, append = TRUE, sep = '|', row.names =
FALSE, col.names = FALSE, quote = FALSE, fileEncoding = "UTF-8")
}

# Create a table mapping the call's cgsccpadd and cdsccpadd to the region or province code

sccp_filenames <- list.files("./sccp1", pattern = "*.csv.gz", full.names = TRUE)
sccp_out_file <- 'sccp_out_sccp0.txt'
headers <- NULL

# Convert the table and only some useful fields are kept

headers <- rbind(headers, c("Endtime", "StartTime", "MS", "OPC", "UserError", "CgAddress", "IMSI",
"Area", "Region", "Carrier", "Network", "Province", "CdAddress", "AreaDest", "RegionDest",
"CarrierDest", "NetworkDest", "Province_Dest"))
write.table(as.data.frame(headers), file = sccp_out_file, append = FALSE, sep = '|', row.names = FALSE,
col.names = FALSE, quote = FALSE)
for (i in seq_along(sccp_filenames)) {
  f <- read.csv(sccp_filenames[i], header = TRUE)
  for (j in 1:dim(f)[1]) {

    cgaddress <- f[j, 41]
    cdaddress <- f[j, 42]
    if ((cgaddress == '') || (cgaddress == '-') || (cdaddress == '') || (cdaddress == '-')) {
      next
    }

# If the calling party's Cg Sccp Add starts with '86', find its corresponding province

    if (substr(cgaddress, 1, 2) == '86') {
      prov = sccp_prov[[substr(cgaddress, 3, 4)]]
      if (!is.null(prov)) {
        user_error = as.character(f[j, 23])
        imsi = as.character(f[j, 57])
        if (imsi != '-') {
          end_time = as.character(f[j, 1])
          start_time = as.character(f[j, 2])
          ms = as.character(f[j, 3])
          opc = as.character(f[j, 8])

# If the called party's Cd Sccp Add starts with '86', find its corresponding province
```

```
          if (substr(cdaddress, 1, 2) == '86') {
            prov_dest = sccp_prov[[substr(cdaddress, 3, 4)]]
            if (!is.null(prov_dest)) {
              line <- rbind(NULL, c(as.character(f[j, 1]), as.character(f[j, 2]), as.character(f[j, 3]),
as.character(f[j, 8]), as.character(f[j, 23]), as.character(f[j, 41]), as.character(f[j, 57]), '-', '-', '-', '-', prov,
as.character(f[j, 42]), '-', '-', '-', '-', prov_dest))
              write.table(as.data.frame(line), file = sccp_out_file, append = TRUE, sep = "|", row.names =
FALSE, col.names = FALSE, quote = FALSE, fileEncoding = "UTF-8")
              next
            }
          }
          else {
            for (k in 3:11) {
              prefix <- substr(cdaddress, 1, k)

# If the call's cdsccpadd not starting with '86', find its region

              mapping <- sccp_dict[[prefix]]
              if (!is.null(mapping)) {
                line <- rbind(NULL, c(as.character(f[j, 1]), as.character(f[j, 2]), as.character(f[j, 3]),
as.character(f[j, 8]), as.character(f[j, 23]), as.character(f[j, 41]), as.character(f[j, 57]), '-', '-', '-', '-', prov,
as.character(f[j, 42]), mapping, '-'))
                write.table(as.data.frame(line), file = sccp_out_file, append = TRUE, sep = "|", row.names =
FALSE, col.names = FALSE, quote = FALSE, fileEncoding = "UTF-8")
                break
              }
            }
          }
        }
      }
    }
    else {
    for (k in 3:11) {
      prefix <- substring(cgaddress, 1, k)

# If the call's cgsccpadd not starting with '86', find its region

       mapping <- sccp_dict[[prefix]]
       if (!is.null(mapping)) {
         user_error = as.character(f[j, 23])
         imsi = as.character(f[j, 57])
         if (imsi != '-') {
           end_time = as.character(f[j, 1])
           star_time = as.character(f[j, 2])
           ms = as.character(f[j, 3])
           opc = as.character(f[j, 8])

# If the called party's cdsccpadd starts with '86', find its corresponding province
```

```
          if (substr(cdaddress, 1, 2) == '86') {
            prov_dest = sccp_prov[[substr(cdaddress, 3, 4)]]
            if (!is.null(prov_dest)) {
              line <- rbind(NULL, c(as.character(f[j, 1]), as.character(f[j, 2]), as.character(f[j, 3]),
as.character(f[j, 8]), as.character(f[j, 23]), as.character(f[j, 41]), as.character(f[j, 57]), mapping, '-',
as.character(f[j, 42]), '-', '-', '-', '-', prov_dest))
              write.table(as.data.frame(line), file = sccp_out_file, append = TRUE, sep = "|", row.names =
FALSE, col.names = FALSE, quote = FALSE, fileEncoding = "UTF-8")
              break
            }
          }
          else {
            for (kk in 3:11) {
              prefix_dest <- substr(cdaddress, 1, kk)

# If the call's Cd Sccp Add not starting with '86', find its region

              mapping_dest <- sccp_dict[[prefix_dest]]
              if (!is.null(mapping_dest)) {
                line <- rbind(NULL, c(as.character(f[j, 1]), as.character(f[j, 2]), as.character(f[j, 3]),
as.character(f[j, 8]), as.character(f[j, 23]), as.character(f[j, 41]), as.character(f[j, 57]), mapping, '-',
as.character(f[j, 42]), mapping_dest, '-'))

# Save the result into the output file

                write.table(as.data.frame(line), file = sccp_out_file, append = TRUE, sep = "|", row.names =
FALSE, col.names = FALSE, quote = FALSE, fileEncoding = "UTF-8")
                break
              }
            }
            break
          }
        }
      }
    }
    }
    }
  }
}
```

3. 汇总每个 OPC 跟 DPC 有效组合的成功与失败连接数目， 可以籍此发现有关信号异常现象：

```
filenames <- list.files("sccp", pattern = "*.csv", full.names = TRUE)
library(plyr)
library(dostats)
```

```
outFile <- "features.csv"

# Count the number of connection, successful and failed connection

outFileColNames <- c("DPC", "OPC", "Count", "Successful", "Failed", "Denyed")
combin <- matrix(0, 15000, 15000)
winner <- matrix(0, 15000, 15000)
failed <- matrix(0, 15000, 15000)
denyed <- matrix(0, 15000, 15000)
for (i in seq_along(filenames)) {
  f <- read.csv(filenames[i])
  for (j in seq(dim(f)[1])) {
    combin[f[j, 7], f[j, 8]] = combin[f[j, 7], f[j, 8]] + 1
    if (f[j, 20] == "Yes") {
      winner[f[j, 7], f[j, 8]] = winner[f[j, 7], f[j, 8]] + 1
    }
    else {
      failed[f[j, 7], f[j, 8]] = failed[f[j, 7], f[j, 8]] + 1
    }

# Count the number of records with User error == Roaming not allowed

    if (f[j, 23] == "Roaming not allowed") {
      denyed[f[j, 7], f[j, 8]] = denyed[f[j, 7], f[j, 8]] + 1
    }
  }
}
result <- NULL
for (m in seq(15000)) {
  for (n in seq(15000)) {
    if (combin[m, n] > 0) {
      result <- rbind(result, c(m, n, combin[m, n], winner[m, n], failed[m, n], denyed[m, n]))
    }
  }
}
write.table(result,file=outFile,sep=',',row.names=FALSE,col.names=outFileColNames,quote=FALSE)
```

利用 HIVE 进行数据分析主要步骤及代码：

假设远程主机地址为 10.32.42.204

1. Ssh 进入远程主机并创建目录 /home/sccp
2. 进入本地主机，并将压缩后数据从本地盘复制到远程地址

   scp *.tar root@10.32.42.204/home/sccp

3. Ssh 进入远程主机，unrar 数据文件(如果没有安装 unrar,请先安装)

unrar e sccp1.rar

如果 rar 文件有多个部分，只需要指定第一部分，unrar 会把所有文件解压完成。整个过程会产生很多.csv 文件

4. 创建 HIVE 准备表. sccp 表有上百个字段，这里只展示了前几个。

```
create table sccp_raw_stage (endtime string, begintime string, ...
) row format delimited fileds terminated by '|' stored as textfile;
```

5.将准备表转换为 ORC 表以优化性能

```
create table sccp_raw stored as orc as SELECT * FROM sccp_raw_stage;
```

7.Sccp 里面的时间表示是非标准化的，需将其标准化，并只保留有用的字段

```
create table sccp_raw_simplified stored as orc as SELECT
from_unixtime(unix_timestamp(begintime, "dd/MM/yyyy h:m:s")) as begintime,
from_unixtime(unix_timestamp(endtime, "dd/MM/yyyy h:m:s")) as endtime,
ms, usererror, cgsccpadd, cdsccpadd, imsi from sccp_raw;
```

8.sccp 中国记录的所在省份是通过代码形式体现的。 首先创建省份查找的准备表

```
create table sccp_province_look_up_stage (code string, province string) row format
delimited fields terminated by '|' stored as textfile;
```

7. 将准备表转换为 ORC 表以优化性能

```
create table sccp_province_look_up stored as orc as select  * from
sccp_province_look_up_stage;
```

8. 根据 cgsccpadd 进行省份的查找。 cgsccpdadd 是主叫方的标识符。通过匹配 86 前缀来识别主叫方来自国内的通话记录

```
create table cn_cg stored as orc as select sccp.*,
lookup.mscid as mscid, lookup.province as province
from sccp_raw_simplified as sccp join new_lookup as lookup
where substring(sccp.cgsccpadd, 1, 2) = "86" and length(imsi) = 15 and
```

substring(sccp.cgsccpadd, 3, 7) = lookup.mscid;

9.创建总结表 。 对于每个 imsi, 得到它所对应的省份的数量

create table cn_cg_summary stored as orc as select imsi, count (distinct province) as cnt from cn_cg group by imsi order by cnt desc;

10. 根据 cdsccpadd 进行省份的查找。 cdsccpadd 是被叫方的标识符。通过匹配 86 前缀来识别被叫方在国内的通话记录

create table cn_cd stored as orc as select sccp.*,

lookup.mscid as mscid, lookup.province as province

from sccp_raw_simplified as sccp join new_lookup as lookup

where substring(sccp.cdsccpadd, 1, 2) = "86" and length(imsi) = 15 and

substring(sccp.cdsccpadd, 3, 7) = lookup.mscid;

11.创建总结表。 对于每个 imsi,得到它所对应的省份的数量

create table cn_cd_summary stored as orc as select imsi, count (distinct province) as cnt from cn_cd group by imsi order by cnt desc;

12. Sccp 中国以外的国家的通话记录是通过代码形式体现的， 首先建立国家查找的的准备表

Create table CREATE TABLE sccp_region_lookup_stage (Prefix string, Area string, Region string, Carrier string, Network string)ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;

13. 将准备表转换为 ORC 表以优化性能

create table sccp_region_lookup stored as orc as select * from sccp_region_lookup_stage;

14. 根据 cgsccpadd 进行国家的查找。 cgsccpdadd 是主叫方的标识符。通过匹配非 86 前缀来识别主叫方来自国外的通话记录。 由于国家前缀长度从 3 到 11 不等， 因此逐个进行匹配。

create table en_cg stored as orc as select sccp.*, lookup.prefix as prefix, lookup.area as area, lookup.region as region, lookup.carrier as carrier, lookup.network as network from sccp_raw_simplified_small as sccp join sccp_region_lookup as lookup where substring(sccp.cgsccpadd, 1, 2) != "86" and (substring(sccp.cgsccpadd, 1, 3) = lookup.prefix or substring(sccp.cgsccpadd, 1, 4) = lookup.prefix or substring(sccp.cgsccpadd, 1, 5) = lookup.prefix or substring(sccp.cgsccpadd, 1, 6) = lookup.prefix or substring(sccp.cgsccpadd, 1, 7) = lookup.prefix or substring(sccp.cgsccpadd, 1, 8) = lookup.prefix or

substring(sccp.cgsccpadd, 1, 9) = lookup.prefix or substring(sccp.cgsccpadd, 1, 10) = lookup.prefix or substring(sccp.cgsccpadd, 1, 11) = lookup.prefix);

15. 创建总结表。 对于每个 imsi,得到它所对应的国家的数量

CREATE TABLE en_cg_summary STORED AS orc AS SELECT region, COUNT(DISTINCT imsi) AS cnt FROM en_cg GROUP BY region ORDER BY cnt DESC;

16. 根据 cdsccpadd 进行国家的查找。 cdsccpdadd 是被叫方的标识符。通过匹配非 86 前缀来识别被叫方来自国外的通话记录。 由于国家前缀长度从 3 到 11 不等， 因此逐个进行匹配

create table en_cd stored as orc as select sccp.*, lookup.prefix as prefix, lookup.area as area, lookup.region as region, lookup.carrier as carrier, lookup.network as network from sccp_raw_simplified_small as sccp join sccp_region_lookup as lookup where substring(sccp.cdsccpadd, 1, 2) != "86" and (substring(sccp.cdsccpadd, 1, 3) = lookup.prefix or substring(sccp.cdsccpadd, 1, 4) = lookup.prefix or substring(sccp.cdsccpadd, 1, 5) = lookup.prefix or substring(sccp.cdsccpadd, 1, 6) = lookup.prefix or substring(sccp.cdsccpadd, 1, 7) = lookup.prefix or substring(sccp.cdsccpadd, 1, 8) = lookup.prefix or substring(sccp.cdsccpadd, 1, 9) = lookup.prefix or substring(sccp.cdsccpadd, 1, 10) = lookup.prefix or substring(sccp.cdsccpadd, 1, 11) = lookup.prefix);

17. 创建总结表。 对于每个 imsi,得到它所对应的国家的数量

CREATE TABLE en_cd_summary STORED AS orc AS SELECT region, COUNT(DISTINCT imsi) AS cnt FROM en_cd GROUP BY region ORDER BY cnt DESC;

18. 建立所有 imsi 和 region 的联系表

CREATE TABLE imsi_region STORED AS orc AS SELECT imsi, region, COUNT(*) AS cnt FROM en_cg WHERE LENGTH(imsi) = 15 GROUP BY imsi, region ORDER BY imsi, region;

19 建立所有 imsi 和 region 的 错误联系表

CREATE TABLE imsi_region_usererror STORED AS orc AS SELECT imsi, region, usererror, COUNT(*) AS cnt FROM en_cg WHERE LENGTH(imsi) = 15 GROUP BY imsi, region, usererror ORDER BY imsi, region, usererror;

20 将各个 imsi， region 对的错误数进行汇总

create TABLE imsi_region_usererror_all stored as orc as select imsi, region, sum(cnt) as cnt from imsi_region_usererror group by imsi,region order by imsi, region;

21 将各个 imsi, region 对中的正确记录数进行汇总

create TABLE imsi_region_usererror_all stored as orc as select imsi, region, sum(cnt) as cnt from imsi_region_usererror group by imsi,region order by imsi, region where usererror = '-'

22 将各个 imsi, region 对中的错误记录数进行汇总

create TABLE imsi_region_usererror_all stored as orc as select imsi, region, sum(cnt) as cnt from imsi_region_usererror group by imsi,region order by imsi, region where usererror = '-'

23 根据正确率和错误率进行分类

create TABLE imsi_region_usererror_sxs stored as orc as select total.imsi, total.region, total.cnt as total_cnt , error.cnt as error_cnt, success.cnt as success_cnt , error.cnt / total.cnt as error_rate, success.cnt / total.cnt as success_rate, case when isnull(error.cnt) then 'all_success' when isnull(success.cnt) then 'all_error' else 'some_error' end as error_class from imsi_region_usererror_all as total left join imsi_region_usererror_error as error on total.imsi = error.imsi and total.region = error.region left join imsi_region_usererror_success as success on total.imsi = success.imsi and total.region = success.region;