Exercise 2

191820019 陈文杰

2020/10/16

R Markdown

#例1. flights数据集中有origin和dest这两个变量，分别表示飞机起飞得往返地；：

1、统计一下，共有多少个不同的往返地组合，并将这些往返地组合抽取出来，构造成一个名为Ori.Dest的新数据集。

解:

##[策略] ①利用group_by函数按起始地分组，并取出origin和dest两列 ②利用unique函数去重

##[过程|结果]

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(nycflights13)

Orin.Dest <- dplyr::group_by(flights,origin,dest)
Orin.Dest <- Orin.Dest[,c('origin','dest')]
unique(Orin.Dest)
```

origin <chr>	**dest** <chr>
EWR	IAH
LGA	IAH
JFK	MIA
JFK	BQN
LGA	ATL
EWR	ORD
EWR	FLL

origin <chr>	dest <chr>
LGA	IAD
JFK	MCO
LGA	ORD

1-10 of 224 rows Previous **1** 2 3 4 5 6 ... 23 Next

2、airports数据集当中，详细的记录着每个机场的经纬度信息。请将flights数据集进行扩充，增加4列，这4列分别是Ori.Lat, Ori.Lon, Dest.Lat, Dest.Lon，分别表示这起飞地的经纬度和到达地的经纬度。使用merge函数，熟悉此函数的不同用途

解:

##[策略] ①利用select从airports中选出faa,lat,lon列 ②利用merge函数合并airports.selected与flights，以faa和origin列作为合并的依据,得到flights2

##[过程|结果]

```
head(airports)
```

…	name <chr⌅chr>	lat <dbl>	lon <dbl>	alt <dbl>	tz <dbl⌅chr⌅chr>	…	tzone
04G	Lansdowne Airport	41.13047	-80.61958	1044	-5	A	America/New_York
06A	Moton Field Municipal Airport	32.46057	-85.68003	264	-6	A	America/Chicago
06C	Schaumburg Regional	41.98934	-88.10124	801	-6	A	America/Chicago
06N	Randall Airport	41.43191	-74.39156	523	-5	A	America/New_York
09J	Jekyll Island Airport	31.07447	-81.42778	11	-5	A	America/New_York
0A9	Elizabethton Municipal Airport	36.37122	-82.17342	1593	-5	A	America/New_York

6 rows

```
colnames(airports)
```

```
## [1] "faa"   "name"  "lat"   "lon"   "alt"   "tz"    "dst"   "tzone"
```

```
colnames(flights)
```

```
##  [1] "year"           "month"          "day"            "dep_time"
##  [5] "sched_dep_time" "dep_delay"      "arr_time"       "sched_arr_time"
##  [9] "arr_delay"      "carrier"        "flight"         "tailnum"
## [13] "origin"         "dest"           "air_time"       "distance"
## [17] "hour"           "minute"         "time_hour"
```

```
head(flights)
```

y... <int>	mo... <int>	... <int>	dep_time <int>	sched_dep_time <int>	dep_delay <dbl>	arr_time <int>	sched_arr_time <int>	arr_delay <dbl>	ca <c
2013	1	1	517	515	2	830	819	11	UA
2013	1	1	533	529	4	850	830	20	UA
2013	1	1	542	540	2	923	850	33	AA
2013	1	1	544	545	-1	1004	1022	-18	B6
2013	1	1	554	600	-6	812	837	-25	DL
2013	1	1	554	558	-4	740	728	12	UA

6 rows | 1-10 of 19 columns

```
airports.selected <- dplyr::select(airports,faa,lat,lon)
flights1 <- merge(flights,airports.selected,by.x=names(flights)[13],by.y=names(airports.selecte
d)[1])
flights1 <- rename(flights1,c(Ori.Lat=lat, Orin.Lon=lon))
flights2 <- merge(flights1,airports.selected,by.x=names(flights)[14],by.y=names(airports.selecte
d)[1])
flights2 <- rename(flights2,c(Dest.Lat=lat, Dest.Lon=lon))
str(flights2)
```

```
## 'data.frame':    329174 obs. of  23 variables:
##  $ dest          : chr  "ABQ" "ABQ" "ABQ" "ABQ" ...
##  $ origin        : chr  "JFK" "JFK" "JFK" "JFK" ...
##  $ year          : int  2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
##  $ month         : int  11 4 11 11 4 7 7 8 10 6 ...
##  $ day           : int  1 22 24 6 27 31 21 15 30 1 ...
##  $ dep_time      : int  1950 1712 2000 1952 2020 2002 2029 2006 1954 1957 ...
##  $ sched_dep_time: int  2000 1630 2000 2000 2025 2007 2007 2007 2000 2001 ...
##  $ dep_delay     : num  -10 42 0 -8 -5 -5 22 -1 -6 -4 ...
##  $ arr_time      : int  2226 1946 2252 2309 2245 2219 2317 2230 2329 2238 ...
##  $ sched_arr_time: int  2303 1915 2303 2303 2304 2259 2259 2259 2303 2308 ...
##  $ arr_delay     : num  -37 31 -11 6 -19 -40 18 -29 26 -30 ...
##  $ carrier       : chr  "B6" "B6" "B6" "B6" ...
##  $ flight        : int  65 1505 65 65 1505 1505 1505 1505 65 1505 ...
##  $ tailnum       : chr  "N659JB" "N821JB" "N633JB" "N661JB" ...
##  $ air_time      : num  253 256 263 279 246 237 248 241 306 239 ...
##  $ distance      : num  1826 1826 1826 1826 1826 ...
##  $ hour          : num  20 16 20 20 20 20 20 20 20 20 ...
##  $ minute        : num  0 30 0 0 25 7 7 7 0 1 ...
##  $ time_hour     : POSIXct, format: "2013-11-01 20:00:00" "2013-04-22 16:00:00" ...
##  $ Ori.Lat       : num  40.6 40.6 40.6 40.6 40.6 ...
##  $ Orin.Lon      : num  -73.8 -73.8 -73.8 -73.8 -73.8 ...
##  $ Dest.Lat      : num  35 35 35 35 35 ...
##  $ Dest.Lon      : num  -107 -107 -107 -107 -107 ...
```

3.构造一个名为Calculate_Distance的函数，该函数可以传递4个参数（上题的4个参数），根据这4个参数，计算出起飞地点和到达地点之间的距离。

4.为flights增加一列，名为Cal.Distance，这一新变量是上题计算出的起飞地和到达地之间的距离。

解:

##[策略] ①利用geosphere包的distm（）函数，根据两点经纬度计算距离 ②循环遍历计算结果

##[过程|结果]

```
library(geosphere)
Calculate_Distance <- function(Orin.Lon,Orin.Lat,Dest.Lon,Dest.Lat){
  muer.lonlat = rbind(t1=c(Orin.Lon,Orin.Lat),t2=c(Dest.Lon,Dest.Lat))
  muer.dists = distm(muer.lonlat, fun=distVincentyEllipsoid)[1,2]
  return (muer.dists)
}

#显示循环
flights2$Cal.Distance <- 'Na'
for (i in 1:dim(flights2)[1])
  {
    flights2[i,'Cal.Distance'] <- Calculate_Distance(flights2[i,21],flights2[i,20],flights2[i,23
],flights2[i,22])
  }
head(flights2$Cal.Distance)
```

```
## [1] "2937884.24662709" "2937884.24662709" "2937884.24662709" "2937884.24662709"
## [5] "2937884.24662709" "2937884.24662709"
```

#例2.planes数据中，有每架飞机的tailnum（机尾编号），可作为飞机唯一标识符，完成下列问题： 1.创建一个名为flights.Planes.Info的新data.frame，其中记录不同飞机的飞行次数、平均飞行距离、平均延误时间，并按照平均延误时间从大到下排序。（使用dplyr包中的一系列函数）

解:

##[策略]

- 利用group_by函数按tailnum进行分组
- 利用na.omit去除空值
- 利用summarise函数分别计算飞行次数、平均距离和平均延误
- 利用arrange函数将summarise所得的数据按平均延误的降序排列

##[过程|结果]

```
library(dplyr)
#分组
tail.num <- group_by(flights2,tailnum)
str(tail.num)
```

```
## tibble [329,174 x 24] (S3: grouped_df/tbl_df/tbl/data.frame)
##  $ dest          : chr [1:329174] "ABQ" "ABQ" "ABQ" "ABQ" ...
##  $ origin        : chr [1:329174] "JFK" "JFK" "JFK" "JFK" ...
##  $ year          : int [1:329174] 2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
##  $ month         : int [1:329174] 11 4 11 11 4 7 7 8 10 6 ...
##  $ day           : int [1:329174] 1 22 24 6 27 31 21 15 30 1 ...
##  $ dep_time      : int [1:329174] 1950 1712 2000 1952 2020 2002 2029 2006 1954 1957 ...
##  $ sched_dep_time: int [1:329174] 2000 1630 2000 2000 2025 2007 2007 2007 2000 2001 ...
##  $ dep_delay     : num [1:329174] -10 42 0 -8 -5 -5 22 -1 -6 -4 ...
##  $ arr_time      : int [1:329174] 2226 1946 2252 2309 2245 2219 2317 2230 2329 2238 ...
##  $ sched_arr_time: int [1:329174] 2303 1915 2303 2303 2304 2259 2259 2259 2303 2308 ...
##  $ arr_delay     : num [1:329174] -37 31 -11 6 -19 -40 18 -29 26 -30 ...
##  $ carrier       : chr [1:329174] "B6" "B6" "B6" "B6" ...
##  $ flight        : int [1:329174] 65 1505 65 65 1505 1505 1505 1505 65 1505 ...
##  $ tailnum       : chr [1:329174] "N659JB" "N821JB" "N633JB" "N661JB" ...
##  $ air_time      : num [1:329174] 253 256 263 279 246 237 248 241 306 239 ...
##  $ distance      : num [1:329174] 1826 1826 1826 1826 1826 ...
##  $ hour          : num [1:329174] 20 16 20 20 20 20 20 20 20 20 ...
##  $ minute        : num [1:329174] 0 30 0 0 25 7 7 7 0 1 ...
##  $ time_hour     : POSIXct[1:329174], format: "2013-11-01 20:00:00" "2013-04-22 16:00:00" ...
##  $ Ori.Lat       : num [1:329174] 40.6 40.6 40.6 40.6 40.6 ...
##  $ Orin.Lon      : num [1:329174] -73.8 -73.8 -73.8 -73.8 -73.8 ...
##  $ Dest.Lat      : num [1:329174] 35 35 35 35 35 ...
##  $ Dest.Lon      : num [1:329174] -107 -107 -107 -107 -107 ...
##  $ Cal.Distance  : chr [1:329174] "2937884.24662709" "2937884.24662709" "2937884.24662709" "2
937884.24662709" ...
##  - attr(*, "groups")= tibble [4,044 x 2] (S3: tbl_df/tbl/data.frame)
##   ..$ tailnum: chr [1:4044] "D942DN" "N0EGMQ" "N10156" "N102UW" ...
##   ..$ .rows  : list<int> [1:4044]
##   .. ..$ : int [1:4] 2642 11122 186494 191914
##   .. ..$ : int [1:371] 1120 1355 2481 2532 2703 3480 3541 3673 3930 4018 ...
##   .. ..$ : int [1:153] 7770 13924 15293 20882 20885 21039 21143 22979 23962 24409 ...
##   .. ..$ : int [1:48] 61944 62285 62474 62724 62873 62893 63825 64277 64833 65015 ...
##   .. ..$ : int [1:46] 62212 62775 62835 62839 63013 63213 63399 63753 63767 63777 ...
##   .. ..$ : int [1:47] 62165 62177 62346 63175 63569 63596 63789 64483 65091 66180 ...
##   .. ..$ : int [1:289] 752 773 791 809 826 833 883 937 4588 6969 ...
##   .. ..$ : int [1:45] 61827 61913 62587 63119 63279 63358 63419 64109 65107 65675 ...
##   .. ..$ : int [1:41] 62039 62174 62229 62274 62362 62626 63762 63868 64046 64341 ...
##   .. ..$ : int [1:60] 61852 62021 62224 62234 62667 62722 62771 62797 63907 64333 ...
##   .. ..$ : int [1:48] 61814 61919 62186 62472 62564 62752 63262 63378 63651 64092 ...
##   .. ..$ : int [1:40] 62313 62810 63075 63897 64590 64630 64786 65515 65611 65893 ...
##   .. ..$ : int [1:129] 3837 6946 8445 8736 12787 14067 20863 22929 24959 27315 ...
##   .. ..$ : int [1:148] 10485 10850 16902 21085 22180 23950 26344 27448 44415 45263 ...
##   .. ..$ : int [1:148] 1557 2088 3719 16055 20679 20700 20916 21001 22036 22108 ...
##   .. ..$ : int [1:138] 854 2067 5825 7119 10419 15553 16481 21018 21151 26186 ...
##   .. ..$ : int [1:148] 615 832 841 5787 5962 15227 20989 25029 25492 27529 ...
##   .. ..$ : int [1:154] 3434 12523 15107 15561 20767 20839 25481 25680 26326 26481 ...
##   .. ..$ : int [1:124] 5736 10266 21310 22556 22619 22673 25532 33136 44296 44372 ...
##   .. ..$ : int [1:112] 1203 10184 10765 17972 20733 20803 21064 22738 24210 24501 ...
##   .. ..$ : int [1:157] 1161 3602 3973 6266 20790 21079 21084 22540 26422 44225 ...
##   .. ..$ : int [1:136] 810 21071 22605 23765 46444 47611 52103 52574 54608 54994 ...
##   .. ..$ : int [1:98] 4248 11146 11708 15321 20765 24626 25283 26209 26841 27931 ...
##   .. ..$ : int [1:143] 521 680 901 909 2279 4151 4220 5891 6392 9203 ...
##   .. ..$ : int [1:159] 647 2257 3426 4623 10195 10670 11314 13333 15696 17072 ...
##   .. ..$ : int [1:142] 748 1323 20756 20837 20843 21285 22695 23147 23336 23407 ...
##   .. ..$ : int [1:125] 767 824 4306 7314 9574 15377 20623 20975 24897 26917 ...
##   .. ..$ : int [1:136] 601 918 8246 9255 9654 9860 10406 10856 21049 23027 ...
##   .. ..$ : int [1:133] 554 686 1780 2052 16147 37857 44377 45347 46510 49072 ...
```

```
##   .. ..$ : int [1:140] 638 860 16935 20906 21154 21218 21244 22087 22388 49744 ...
##   .. ..$ : int [1:136] 808 1244 5759 16715 20761 20773 21299 24852 45507 50984 ...
##   .. ..$ : int [1:154] 11550 17136 20706 21028 21131 46072 46075 46957 52661 53419 ...
##   .. ..$ : int [1:161] 533 709 787 6673 14247 17932 21074 21298 24132 24139 ...
##   .. ..$ : int [1:156] 788 1924 9207 12388 16711 20643 20699 20800 21178 24285 ...
##   .. ..$ : int [1:126] 733 940 1272 3651 3737 3889 6906 9243 17043 21260 ...
##   .. ..$ : int [1:30] 61976 62270 62439 64236 64472 64715 65292 65698 65910 66025 ...
##   .. ..$ : int [1:109] 18541 18898 19377 20121 28562 29211 30436 31733 32218 33264 ...
##   .. ..$ : int [1:38] 61975 62488 62525 63243 63247 63562 63848 64824 65043 65128 ...
##   .. ..$ : int [1:43] 62812 63814 63932 64127 64668 64877 64937 65842 66467 67192 ...
##   .. ..$ : int [1:39] 38916 61910 61993 62096 62154 62404 62693 63081 63157 63683 ...
##   .. ..$ : int [1:232] 675 4157 6703 8037 8646 9649 10161 11381 15010 15043 ...
##   .. ..$ : int [1:277] 678 922 2467 4628 5233 7930 8678 16841 20641 20852 ...
##   .. ..$ : int [1:229] 581 696 3551 5817 6126 6568 9025 9242 12985 18116 ...
##   .. ..$ : int [1:226] 584 668 731 821 2426 2584 9435 10938 11435 11774 ...
##   .. ..$ : int [1:266] 734 5593 8087 8562 8582 8679 9535 11790 14182 15768 ...
##   .. ..$ : int [1:247] 671 689 708 859 2545 5729 7821 9159 9839 11164 ...
##   .. ..$ : int [1:242] 851 1196 1594 2903 5377 6001 6201 7752 8026 9505 ...
##   .. ..$ : int [1:243] 621 811 914 958 1637 2449 2675 3005 5240 5457 ...
##   .. ..$ : int [1:46] 61862 61961 62087 62310 62605 63832 64509 64582 64639 64998 ...
##   .. ..$ : int [1:43] 61881 61951 62055 62145 62471 62548 62850 63021 63906 63975 ...
##   .. ..$ : int [1:39] 62412 62869 63225 63312 63543 63800 64132 64692 65481 65635 ...
##   .. ..$ : int [1:21] 1968 5980 9821 13566 16851 166814 166929 168435 171676 172518 ...
##   .. ..$ : int [1:15] 1628 7500 8417 10986 13862 14889 165686 168210 171833 173081 ...
##   .. ..$ : int [1:85] 94928 94991 95469 97330 97647 101620 145462 148519 148758 149577 ...
##   .. ..$ : int [1:68] 36680 43557 96183 96320 98710 99066 99711 101749 127712 144960 ...
##   .. ..$ : int [1:77] 96387 96624 98168 98585 98952 99044 100366 145049 145556 146976 ...
##   .. ..$ : int [1:128] 698 2876 7061 16072 20762 23596 25145 26427 44026 44087 ...
##   .. ..$ : int [1:92] 97479 98196 98771 100197 100658 100768 123635 145016 146788 146816 ...
##   .. ..$ : int [1:152] 834 968 11624 14647 15980 20868 20988 22558 23366 24185 ...
##   .. ..$ : int [1:143] 525 750 1453 2594 6402 6957 8307 11913 18114 20665 ...
##   .. ..$ : int [1:123] 564 1699 5043 5735 6464 11136 13677 14356 17047 20798 ...
##   .. ..$ : int [1:129] 899 1818 11577 20656 24081 25670 45486 46307 48850 50207 ...
##   .. ..$ : int [1:85] 20720 20732 20766 23069 26298 45314 46279 46657 54999 55108 ...
##   .. ..$ : int [1:144] 2884 3147 7619 7998 10272 20940 22749 23302 24940 26184 ...
##   .. ..$ : int [1:116] 793 5241 5801 6644 6645 21255 23108 24258 24472 28200 ...
##   .. ..$ : int [1:137] 677 6587 7198 11791 22065 23669 24325 26222 44611 44948 ...
##   .. ..$ : int [1:125] 715 861 890 905 17699 21147 25345 25577 27734 44969 ...
##   .. ..$ : int [1:156] 670 4665 6672 7300 7323 10151 17199 18156 20878 21053 ...
##   .. ..$ : int [1:132] 8723 8986 14686 23018 24423 26106 36574 43859 44383 45781 ...
##   .. ..$ : int [1:171] 777 804 823 6656 7097 7837 8198 17480 20853 21127 ...
##   .. ..$ : int [1:148] 569 625 633 2457 12433 12458 16388 20883 20937 23907 ...
##   .. ..$ : int [1:2] 9262 9998
##   .. ..$ : int [1:31] 62105 62127 62353 62462 63203 63405 63706 63820 64266 64273 ...
##   .. ..$ : int [1:130] 649 665 827 1315 20978 21096 26187 45194 46211 49275 ...
##   .. ..$ : int [1:102] 19043 19923 28809 30394 33470 36648 38241 58444 58610 59383 ...
##   .. ..$ : int [1:109] 11982 30371 32180 32588 35930 36172 37671 37742 39031 41560 ...
##   .. ..$ : int [1:120] 18329 18546 18557 19466 20221 28608 33001 33119 35507 36824 ...
##   .. ..$ : int [1:109] 19160 29367 29507 30437 33287 34790 35312 38589 40083 41258 ...
##   .. ..$ : int [1:92] 18687 34256 34270 36192 37006 58053 75381 100757 106516 109837 ...
##   .. ..$ : int [1:45] 62013 62027 62069 62081 62435 62663 62727 64203 64292 65224 ...
##   .. ..$ : int [1:33] 62110 62171 64776 64870 65578 66097 66711 66899 66992 67125 ...
##   .. ..$ : int [1:46] 62265 62487 62540 63451 63578 64371 64566 64798 65120 65623 ...
##   .. ..$ : int [1:225] 738 756 758 845 886 893 7715 8960 9174 10423 ...
##   .. ..$ : int [1:244] 608 3890 11878 13244 15002 15616 20707 20738 21120 21306 ...
##   .. ..$ : int [1:202] 523 585 924 7008 13917 20941 21098 21201 21231 21252 ...
##   .. ..$ : int [1:250] 695 707 711 737 836 871 872 4753 5575 5579 ...
##   .. ..$ : int [1:274] 545 794 946 2239 3131 5898 6112 6243 7867 10822 ...
```

```
##   .. ..$ : int [1:237] 730 856 904 3364 5573 6923 7213 12532 15355 20662 ...
##   .. ..$ : int [1:35] 61841 61986 62106 62672 63030 63128 63165 63754 63947 64522 ...
##   .. ..$ : int [1:44] 61896 62309 62469 63118 63888 64380 64879 64944 65101 65367 ...
##   .. ..$ : int [1:42] 61963 62303 62768 63275 63449 64047 64094 64548 64608 64637 ...
##   .. ..$ : int [1:39] 62259 63057 63527 63843 64647 64737 65428 65602 66046 66079 ...
##   .. ..$ : int [1:237] 606 798 884 1140 2757 5369 6974 8940 11832 15420 ...
##   .. ..$ : int [1:280] 555 1069 5837 6004 6320 6719 7239 8213 9258 10994 ...
##   .. ..$ : int [1:243] 560 848 1050 3518 4728 5217 5485 7041 8436 8610 ...
##   .. ..$ : int [1:245] 796 951 3467 6613 8199 8398 13013 13546 16849 20701 ...
##   .. ..$ : int [1:220] 673 862 1007 1108 1154 6132 16083 20629 20752 20994 ...
##   .. ..$ : int [1:230] 661 927 2471 9426 11150 15209 16239 20763 20902 20950 ...
##   .. ..$ : int [1:261] 676 843 880 894 912 6405 9374 12122 14174 15034 ...
##   .. .. [list output truncated]
##   .. ..@ ptype: int(0)
##   ..- attr(*, ".drop")= logi TRUE
```

```
#去除空值
na.omit(tail.num)
```

d…	origin	y…	mo…	…	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time
<chr>	<chr>	<int>	<int>	<int>	<int>	<int>	<dbl>	<int>	<int>
ABQ	JFK	2013	11	1	1950	2000	-10	2226	2303
ABQ	JFK	2013	4	22	1712	1630	42	1946	1915
ABQ	JFK	2013	11	24	2000	2000	0	2252	2303
ABQ	JFK	2013	11	6	1952	2000	-8	2309	2303
ABQ	JFK	2013	4	27	2020	2025	-5	2245	2304
ABQ	JFK	2013	7	31	2002	2007	-5	2219	2259
ABQ	JFK	2013	7	21	2029	2007	22	2317	2259
ABQ	JFK	2013	8	15	2006	2007	-1	2230	2259
ABQ	JFK	2013	10	30	1954	2000	-6	2329	2303
ABQ	JFK	2013	6	1	1957	2001	-4	2238	2308

1-10 of 10,000 rows | 1-10 of 24 columns Previous **1** 2 3 4 5 6 ... 1000 Next

```
#统计描述
summarise(tail.num,
          count.times = length(tailnum),
          meanDistance = mean(tail.num$distance),
          meanDelay = mean(arr_time-sched_arr_time)
          )
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

tailnum	count.times	meanDistance	meanDelay
<chr>	<int>	<dbl>	<dbl>
D942DN	4	1026.976	6.150000e+01

tailnum <chr>	count.times <int>	meanDistance <dbl>	meanDelay <dbl>
N0EGMQ	371	1026.976	*NA*
N10156	153	1026.976	*NA*
N102UW	48	1026.976	-3.958333e-01
N103US	46	1026.976	-1.910870e+01
N104UW	47	1026.976	*NA*
N10575	289	1026.976	*NA*
N105UW	45	1026.976	-1.004444e+01
N107US	41	1026.976	-1.743902e+01
N108UW	60	1026.976	-8.583333e+00

1-10 of 4,044 rows Previous **1** 2 3 4 5 6 ... 405 Next

```
#排序
flights.Planes.Info <-arrange(
  summarise(tail.num,
            count.times = length(tailnum),
            meanDistance = mean(tail.num$distance),
            meanDelay = mean(arr_time-sched_arr_time)
  ),
  desc(meanDelay)
)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
flights.Planes.Info
```

tailnum <chr>	count.times <int>	meanDistance <dbl>	meanDelay <dbl>
N911DA	1	1026.976	4.940000e+02
N922EV	1	1026.976	4.360000e+02
N587NW	1	1026.976	4.240000e+02
N851NW	1	1026.976	3.790000e+02
N928DN	1	1026.976	3.610000e+02
N7715E	1	1026.976	3.080000e+02
N427SW	1	1026.976	2.770000e+02
N790SK	1	1026.976	2.600000e+02
N136DL	1	1026.976	2.260000e+02
N78003	1	1026.976	2.170000e+02

1-10 of 4,044 rows Previous **1** 2 3 4 5 6 ... 405 Next

2.planes数据集中有一个变量叫manufacturer，表示该架飞机的制造商。请根据flights中的飞行次数，挑选出飞行次数最多的5家飞机制造商。

##[策略]

- 按航班号合并数据
- 利用arrange按航班飞行次数排序
- 利用索引法去重
- 选出排行前五的制造商

##[过程|结果]

```
#按航班号合并数据集
plan.flight <- merge(planes,flights.Planes.Info,by='tailnum')
#排序、去重、访问
flight.sorted <-arrange(plan.flight,desc(count.times))
index <- duplicated(flight.sorted$manufacturer)
flight.sorted <- as.data.frame(flight.sorted)
flight.sorted <- flight.sorted[!index,]
manu <- flight.sorted[1:5,'manufacturer']
manu
```

```
## [1] "GULFSTREAM AEROSPACE" "EMBRAER"              "BOEING"
## [4] "BOMBARDIER INC"       "AIRBUS"
```

3.在问题2的基础上，将flights数据集中这5家飞机制造商的相关飞行记录提取出来，构造一个名为flights.Top5的data.frame。

##[策略]

- 利用merge根据tailnum合并flights和planes数据集
- 利用group_by按照制造商分组
- 利用索引法去重
- 利用filter函数针对前五制造商来过滤选取相应的飞行记录

##[过程|结果]

```
knitr::opts_chunk$set(echo = TRUE)
plan.flight2 <- merge(flights,planes,by='tailnum')
plan.flight2 <- group_by(plan.flight2,'manufacturer')
flights.Top5 <- plan.flight2[1,]
for(i in manu)
{
  tmp <- filter(plan.flight2,plan.flight2$manufacturer==i)

  flights.Top5<-rbind(flights.Top5,tmp)
}
flights.Top5
```

tailnum <chr>	year.x <int>	mo… <int>	… <int>	dep_time <int>	sched_dep_time <int>	dep_delay <dbl>	arr_time <int>	sched_arr_time <int>	a
N10156	2013	8	12	1312	1317	-5	1528	1528	
N344AA	2013	2	10	1906	1729	97	2156	2049	
N344AA	2013	7	22	1857	1505	232	2303	1835	

tailnum	year.x	mo…	…	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	a
<chr>	<int>	<int>	<int>	<int>	<int>	<dbl>	<int>	<int>	
N344AA	2013	2	8	1031	1030	1	1338	1355	
N344AA	2013	2	22	812	810	2	925	925	
N344AA	2013	1	16	1727	1729	-2	2031	2049	
N344AA	2013	1	21	2046	1745	181	17	2120	
N344AA	2013	7	6	*NA*	1530	*NA*	*NA*	1855	
N344AA	2013	7	7	1618	1530	48	2037	1855	
N344AA	2013	1	2	1530	1530	0	1859	1910	

1-10 of 10,000 rows | 1-10 of 28 columns Previous **1** 2 3 4 5 6 ... 1000 Next