<데이터 전처리>
(1) 내부데이터 전처리(R 활용)

< Code 1 - 소지역 코드 분류 >[1)]

▶ 코드 설명 : 신도시별 소지역 코드를 분류하기 위한 코드

1) 원데이터 호출 및 경기도 지역 한정

- **소지역 코드 원데이터 호출**

```
> rawdata <- read.csv("소지역 코드.csv", stringsAsFactors = F)
> rawdata$BLOCK_CD <- as.numeric(gsub(",", "", rawdata$BLOCK_CD)) # ","를 ""로 모두 일괄적 처리
```

- **소지역 코드 - 경기도만 추출**

```
> rawdata_g <- subset(rawdata, rawdata$SIDO_NM == "경기도")
```

2) 1기 신도시 : 행정동 확인

- **1. 분당신도시**

- 행정동 : (성남시 분당구) *금곡동, 구미1동
- 법정동 : (성남시 분당구) 야탑동, 이매동, 서현동, 분당동, 수내동, *정자동

- **2. 일산신도시**

- 행정동 : (고양시 일산동구) 장항2동, 백석1동, 백석2동, 정발산동, (고양시 일산서구) 대화동
- 법정동 : (고양시 일산동구) 마두동, (고양시 일산서구) 일산동, 주엽동

- **3. 평촌신도시**

- 행정동 : (안양시 동안구) *갈산동, 귀인동, 범계동, *부림동, 부흥동, *신촌동, 평안동, 평촌동

- **4. 산본신도시**

- 행정동 : (군포시) 산본1동, 산본2동, 재궁동, *오금동, 수리동, *궁내동, 광정동

- **5. 중동신도시**

- 행정동 : (부천시) *중동, 중1동, 중2동, 중3동, 중4동, 심곡3동, 약대동, *신흥동
- 법정동 : *송내동

3) 1기 신도시 : 소지역 코드 분류

※ 위에서 *로 표기한 행정동이나 법정동은 경기도의 다른 시와 겹쳐 해당 시로 한정한 후 소지역 코드를 추출하였음.

- **1. 분당신도시**

(1) 성남시 분당구만 우선 추출

```
> rawdata_g_bundang <- subset(rawdata_g, rawdata_g$SGNG_NM == "성남시 분당구")
```

(2) 해당 행정동과 법정동만 추출

```
> bundang <- subset(rawdata_g_bundang, rawdata_g_bundang$ADONG_NM %in% c("금곡동", "구미1동"))
> bundang <- rbind(bundang, subset(rawdata_g_bundang, rawdata_g_bundang$LDONG_NM %in% c("야탑동",
"이매동", "서현동", "분당동", "수내동", "정자동")))
```

(3) 분당신도시 BLOCK_CD 추출

```
> cd_bundang <- unique(bundang$BLOCK_CD)
```

※ 위와 같은 방법으로 총 5개의 1기 신도시에 대하여 소지역 코드를 분류하였음.

- **2. 일산신도시**

```
> cd_ilsan <- unique(ilsan$BLOCK_CD)
```

- **3. 평촌신도시**

```
> cd_pyeongchon <- unique(pyeongchon$BLOCK_CD)
```

- **4. 산본신도시**

```
> cd_sanbon <- unique(sanbon$BLOCK_CD)
```

- **5. 중동신도시**

```
> cd_jungdong <- unique(jungdong$BLOCK_CD)
```

4) 2기 신도시 : 행정동 확인

- 1. 판교신도시

- 행정동 : (성남시 분당구) 삼평동, 백현동, 판교동, 운중동

- 2. 한강신도시

- 행정동 : (김포시) 장기동, 운양동, 구래동

- 3. 운정신도시

- 행정동 : (파주시) 운정1동, 운정2동, 운정3동

- 4. 광교신도시

- 행정동 : (수원시 영통구) 원천동, 광교1동, 광교2동, (용인시 수지구) 상현1동, 상현2동

- 5. 양주신도시

- 법정동 : (양주시) 옥정동, 회정동

- 6. 동탄신도시

- 행정동 : (화성시) 동탄1동, 동탄2동, 동탄3동, 동탄4동

5) 2기 신도시 : 소지역 코드 분류

- 1. 분당신도시

(1) 해당 행정동과 법정동만 추출

```
> pangyo <- subset(rawdata_g, rawdata_g$ADONG_NM %in% c("삼평동", "백현동", "판교동", "운중동"))
```

(2) 판교신도시 BLOCK_CD 추출

```
> cd_pangyo <- unique(pangyo$BLOCK_CD)
```

※ 위와 같은 방법으로 총 6개의 2기 신도시에 대하여 소지역 코드를 분류하였음.

- 2. 한강신도시

```
> cd_hangang <- unique(hangang$BLOCK_CD)
```

- 3. 운정신도시

```
> cd_unjeong <- unique(unjeong$BLOCK_CD)
```

- 4. 광교신도시

```
> cd_gwanggyo <- unique(gwanggyo$BLOCK_CD)
```

- 5. 양주신도시

```
> cd_yangju <- unique(yangju$BLOCK_CD)
```

- 6. 동탄신도시

```
> cd_dongtan <- unique(dongtan$BLOCK_CD)
```

< Code 2 - 업종별매출 >[2)]

▶ 코드 설명 : 신도시 별로 업종별 매출(교육, 의료)의 평균을 확인하기 위한 코드

▶ Code 1과 연계되므로 Code 1을 실행해야 작동됨.

1) "업종별매출_yymm.csv" 파일명 제작 및 확인

```
> make <- function(y) {
+   filename <- NULL
+   for(i in 1:12) {
+     if(i %in% 1:9) {
+       filename <- c(filename, paste0("업종별매출_", y, 0, i, ".csv"))
+     } else {
+       filename <- c(filename, paste0("업종별매출_", y, i, ".csv"))
+     }
+   }
+   return(filename)
+ }
```

- **2015년 업종별매출 csv file 이름**

```
> file15 <- make(15)
> file15
 [1] "업종별매출_1501.csv" "업종별매출_1502.csv" "업종별매출_1503.csv" "업종별매출_1504.csv" "업종별매출
_1505.csv" "업종별매출_1506.csv" "업종별매출_1507.csv"
 [8] "업종별매출_1508.csv" "업종별매출_1509.csv" "업종별매출_1510.csv" "업종별매출_1511.csv" "업종별매출
_1512.csv"
```

- **2016년 업종별매출 csv file 이름**

```
> file16 <- make(16)
> file16
 [1] "업종별매출_1601.csv" "업종별매출_1602.csv" "업종별매출_1603.csv" "업종별매출_1604.csv" "업종별매출
_1605.csv" "업종별매출_1606.csv" "업종별매출_1607.csv"
 [8] "업종별매출_1608.csv" "업종별매출_1609.csv" "업종별매출_1610.csv" "업종별매출_1611.csv" "업종별매출
_1612.csv"
```

2) 1기 신도시 : 업종별 매출 추출

```
> if(!require(data.table)) {
+   install.packages("data.table")
+ }
> library(data.table)
```

- **1. 분당신도시**

```
> findsale_b <- function(x = file15, y = file16) {
+   ksale <- NULL
+   for(fx in c(x, y)) {
+     a <- fread(fx, stringsAsFactors = F, encoding = "UTF-8")
+     ksale <- rbind(ksale, subset(a, a$BLOCK_CD %in% cd_bundang))
+   }
+   bundang15 <<- subset(ksale, substr(ksale$STD_YM, 1, 4) == "2015")
+   bundang16 <<- subset(ksale, substr(ksale$STD_YM, 1, 4) == "2016")
+ }

> findsale_b()
```

- 2015, 2016년 업종별 매출 데이터 : bundang15, bundang16

※ 위와 같은 방법으로 총 11개의 1, 2기 신도시 모두에 대하여 업종별 매출을 추출하였음.

- **2. 일산신도시**

- 2015, 2016년 업종별 매출 데이터 : ilsan15, ilsan16

- **3. 평촌신도시**

- 2015, 2016년 업종별 매출 데이터 : pyeongchon15, pyeongchon16

- **4. 산본신도시**

- 2015, 2016년 업종별 매출 데이터 : sanbon15, sanbon16

- **5. 중동신도시**

- 015, 2016년 업종별 매출 데이터 : jungdong15, jungdong16

3) 2기 신도시 : 업종별 매출 추출

- **1. 판교신도시**

- 2015, 2016년 업종별매출 데이터 : pangyo15, pangyo16

- **2. 한강신도시**

- 2015, 2016년 업종별매출 데이터 : hangang15, hangang16

- **3. 운정신도시**

- 2015, 2016년 업종별매출 데이터 : unjeong15, unjeong16

- **4. 광교신도시**

- 2015, 2016년 업종별매출 데이터 : gwanggyo15, gwanggyo16

• **5. 양주신도시**
- 2015, 2016년 업종별매출 데이터 : yangju15, yangju16

• **6. 동탄신도시**
- 2015, 2016년 업종별매출 데이터 : dongtan15, dongtan16

< Code 3 - 유동인구 총합 추이 및 남녀 비율 >[3)]

▶ 코드 설명 : 유동인구 총합 추이 및 남녀 비율을 확인하기 위한 코드
▶ (1) 총합 추이 : 연령대, 성별을 고려하지 않고 합침. 전반적인 추이 파악이 주목적임.
▶ (2) 남녀 비율 : 연령대를 고려하지 않고 성별만 고려함.
▶ Code 1과 연계되므로 Code 1을 실행해야 작동됨.

1) "성연령별유동인구_yymm.csv" 파일명 제작 및 확인

```
> makepop <- function(y) {
+   filename <- NULL
+   for(i in 1:12) {
+     if(i %in% 1:9) {
+       filename <- c(filename, paste0("성연령별유동인구_", y, 0, i, ".csv"))
+     } else {
+       filename <- c(filename, paste0("성연령별유동인구_", y, i, ".csv"))
+     }
+   }
+   return(filename)
+ }
```

• **2015년 성연령별유동인구 csv file 이름**

```
> filepop15 <- makepop(15)
> filepop15
 [1] "성연령별유동인구_1501.csv" "성연령별유동인구_1502.csv" "성연령별유동인구_1503.csv" "성연령별유동인구
_1504.csv" "성연령별유동인구_1505.csv"
 [6] "성연령별유동인구_1506.csv" "성연령별유동인구_1507.csv" "성연령별유동인구_1508.csv" "성연령별유동인구
_1509.csv" "성연령별유동인구_1510.csv"
[11] "성연령별유동인구_1511.csv" "성연령별유동인구_1512.csv"
```

• **2016년 성연령별유동인구 csv file 이름**

```
> filepop16 <- makepop(16)
> filepop16
 [1] "성연령별유동인구_1601.csv" "성연령별유동인구_1602.csv" "성연령별유동인구_1603.csv" "성연령별유동인구
_1604.csv" "성연령별유동인구_1605.csv"
 [6] "성연령별유동인구_1606.csv" "성연령별유동인구_1607.csv" "성연령별유동인구_1608.csv" "성연령별유동인구
_1609.csv" "성연령별유동인구_1610.csv"
[11] "성연령별유동인구_1611.csv" "성연령별유동인구_1612.csv"
```

2) 날짜 입력 함수 제작

```
> makedate <- function(x = 15, y = 16) {
+   mydate <<- NULL
+   for(yy in c(x, y)) {
+     for(m in 1:12) {
+       if(m %in% 1:9) {
+         mydate <<- c(mydate, as.numeric(paste0(yy, 0, m)))
+       } else {
+         mydate <<- c(mydate, as.numeric(paste0(yy, m)))
+       }
+     }
+   }
+   mydate <<- as.factor(mydate)
+ }
```

3) 1기 신도시 : 유동인구 총합 추이, 남녀 비율 추출

```
> if(!require(data.table)) {
+   install.packages("data.table")
+ }
> library(data.table)
```

- 1. 분당신도시

```
> findpop_b <- function(x = filepop15, y = filepop16) {
+   kdata <- NULL
+
+   for(fx in c(x, y)) {
+     a <- fread(fx, stringsAsFactors = F)
+     kdata <- rbind(kdata, subset(a, a$BLOCK_CD %in% cd_bundang))
+   }
+
+   kdata$year <- as.numeric(substr(kdata$STD_YM, 1, 4))
+   kdata$month <- as.numeric(substr(kdata$STD_YM, 5, 6))
+         kdata$man   <-   kdata$MAN_FLOW_POP_CNT_10G   +   kdata$MAN_FLOW_POP_CNT_20G   +
kdata$MAN_FLOW_POP_CNT_30G +
+                       kdata$MAN_FLOW_POP_CNT_40G     +     kdata$MAN_FLOW_POP_CNT_50G     +
kdata$MAN_FLOW_POP_CNT_60GU
+       kdata$woman   <-   kdata$WMAN_FLOW_POP_CNT_10G   +   kdata$WMAN_FLOW_POP_CNT_20G   +
kdata$WMAN_FLOW_POP_CNT_30G +
+                     kdata$WMAN_FLOW_POP_CNT_40G     +     kdata$WMAN_FLOW_POP_CNT_50G     +
kdata$WMAN_FLOW_POP_CNT_60GU
+
+   man.fp <- NULL
+   woman.fp <- NULL
+
+   for(y in 2015:2016) {
+     for(m in 1:12) {
+       subdata <- subset(kdata, kdata$year == y & kdata$month == m)
+       man.fp <- c(man.fp, sum(subdata$man))
+       woman.fp <- c(woman.fp, sum(subdata$woman))
+     }
+   }
+
+   bundang.fp <<- data.frame(matrix(data = NA, nrow = 24, ncol = 2))
+   bundang.fp[, 1] <<- mydate
+   for(i in 1:24) {
+     bundang.fp[i, 2] <<- sum(c(man.fp[i], woman.fp[i]))
+   }
+   colnames(bundang.fp) <<- c("date_ym", "pop_total")
+
+   bundang.fp.rat <<- data.frame(matrix(data = NA, nrow = 2, ncol = 3))
+   bundang.fp.rat[, 1] <<- factor(c("man_pop", "woman_pop"))
+   bundang.fp.rat[, 2] <<- c(sum(man.fp), sum(woman.fp))
+   bundang.fp.rat[, 3] <<- c(sum(man.fp), sum(woman.fp))/sum(bundang.fp.rat[, 2])
+   colnames(bundang.fp.rat) <<- c("gender", "gender_pop", "gender_ratio")
+ }

> findpop_b()
```

※ 변수명 설명
- bundang.fp : 유동인구 총합 추이 데이터 프레임
- bundang.fp.rat : 유동인구 남녀 비율 데이터 프레임

```
> str(bundang.fp)
'data.frame':    24 obs. of  2 variables:
 $ date_ym  : Factor w/ 24 levels "1501","1502",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ pop_total: num  1849620 1668606 1897919 1935281 1870637 ...
```

※ 위와 같은 방법으로 총 11개의 1, 2기 신도시 모두에 대하여 총합 추이와 남녀 비율을 추출하였음.
※ 해당 보고서에는 '총합 추이' 결과만 작성하였음.

- 2. 일산신도시

```
> str(ilsan.fp)
'data.frame':    24 obs. of  2 variables:
 $ date_ym  : Factor w/ 24 levels "1501","1502",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ pop_total: num  1583212 1444724 1638595 1723730 1670378 ...
```

- 3. 평촌신도시

```
> str(pyeongchon.fp)
'data.frame':    24 obs. of  2 variables:
 $ date_ym  : Factor w/ 24 levels "1501","1502",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ pop_total: num  875846 796449 911280 932721 911702 ...
```

- 4. 산본신도시

```
> str(sanbon.fp)
'data.frame':    24 obs. of  2 variables:
 $ date_ym  : Factor w/ 24 levels "1501","1502",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ pop_total: num  573191 529831 606419 625622 622551 ...
```

- 5. 중동신도시

```
> str(jungdong.fp)
'data.frame':    24 obs. of  2 variables:
 $ date_ym  : Factor w/ 24 levels "1501","1502",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ pop_total: num  1261358 1150125 1305397 1329870 1290275 ...
```

4) 2기 신도시 : 유동인구 총합 추이, 남녀 비율 추출

- 1. 판교신도시

```
> str(pangyo.fp)
'data.frame':    24 obs. of  2 variables:
 $ date_ym  : Factor w/ 24 levels "1501","1502",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ pop_total: num  517056 467345 566872 600577 556078 ...
```

- 2. 한강신도시

```
> str(hangang.fp)
'data.frame':    24 obs. of  2 variables:
 $ date_ym  : Factor w/ 24 levels "1501","1502",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ pop_total: num  275378 253233 299149 315381 310878 ...
```

- 3. 운정신도시

```
> str(unjeong.fp)
'data.frame':    24 obs. of  2 variables:
 $ date_ym  : Factor w/ 24 levels "1501","1502",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ pop_total: num  453023 420735 494500 514317 508023 ...
```

- 4. 광교신도시

```
> str(gwanggyo.fp)
'data.frame':    24 obs. of  2 variables:
 $ date_ym  : Factor w/ 24 levels "1501","1502",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ pop_total: num  585397 538737 652750 672840 654778 ...
```

- 5. 양주신도시

```
> str(yangju.fp)
'data.frame':    24 obs. of  2 variables:
 $ date_ym  : Factor w/ 24 levels "1501","1502",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ pop_total: num  62162 57276 66562 68229 69000 ...
```

- 6. 동탄신도시

```
> str(dongtan.fp)
'data.frame':    24 obs. of  2 variables:
```

```
 $ date_ym  : Factor w/ 24 levels "1501","1502",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ pop_total: num  700199 629013 766690 783726 748904 ...
```

< Code 5 - 신도시별 상권 >[4)]

▶ 코드 설명 : 신도시별 상권 위치를 파악하기 위한 코드
▶ Code 1과 연계되므로 Code 1을 실행해야 작동됨.

1) 1기 신도시 : 상권 코드 추출

- **1. 분당신도시**

```
> bz_bundang <- unique(bundang$BZ_CD)
> bz_bundang <- subset(bz_bundang, substr(bz_bundang, 2, 2) == "_")
```

※ 위와 같은 방법으로 총 11개의 1, 2기 신도시 모두에 대하여 상권 코드를 추출하였음.

- **2. 일산신도시**

- 상권 코드 : bz_ilsan

- **3. 평촌신도시**

- 상권 코드 : bz_pyeongchon

- **4. 산본신도시**

- 상권 코드 : bz_sanbon

- **5. 중동신도시**

- 상권 코드 : bz_jungdong

2) 2기 신도시 : 상권 코드 추출

- **1. 판교신도시**

- 상권 코드 : bz_pangyo

- **2. 한강신도시**

- 상권 코드 : bz_hangang

- **3. 운정신도시**

- 상권 코드 : bz_unjeong

- **4. 광교신도시**

- 상권 코드 : bz_gwanggyo

- **5. 양주신도시**

- 상권 코드 : bz_yangju

- **6. 동탄신도시**

- 상권 코드 : bz_dongtan

3) 상권 위치 확인을 위한 "1000대 상권.csv" 호출

```
> if(!require(data.table)) {
  install.packages("data.table")
}
> library(data.table)

> saledata <- fread("1000대 상권.csv", stringsAsFactors = F)
```

4) 1기 신도시 : 상권 위치 확인

※ 위에서 추출한 신도시별 상권 코드를 이용함.

```
> findsale_nt1 <- function(x = saledata) {
+   bundang.sale <<- NULL; ilsan.sale <<- NULL; pyeongchon.sale <<- NULL
+   sanbon.sale <<- NULL; jungdong.sale <<- NULL
+
+   bundang.sale <<- rbind(bundang.sale, subset(x, x$BZ_CD %in% bz_bundang))
```

```
+    ilsan.sale <<- rbind(ilsan.sale, subset(x, x$BZ_CD %in% bz_ilsan))
+    pyeongchon.sale <<- rbind(pyeongchon.sale, subset(x, x$BZ_CD %in% bz_pyeongchon))
+    sanbon.sale <<- rbind(sanbon.sale, subset(x, x$BZ_CD %in% bz_sanbon))
+    jungdong.sale <<- rbind(jungdong.sale, subset(x, x$BZ_CD %in% bz_jungdong))
+ }

> findsale_nt1()
```

- **1. 분당신도시**

- 상권 결과 : bundang.sale

- **2. 일산신도시**

- 상권 결과 : ilsan.sale

- **3. 평촌신도시**

- 상권 결과 : pyeongchon.sale

- **4. 산본신도시**

- 상권 결과 : sanbon.sale

- **5. 중동신도시**

- 상권 결과 : jungdong.sale

5) 2기 신도시 : 상권 위치 확인

※ 위의 1기 신도시 findsale_nt1()와 같은 구조로 총 6개의 2기 신도시에 대하여 상권 위치를 추출하였음.

- **1. 판교신도시**

- 상권 결과 : pangyo.sale

- **2. 한강신도시**

- 상권 결과 : hangang.sale

- **3. 운정신도시**

- 상권 결과 : unjeong.sale

- **4. 광교신도시**

- 상권 결과 : gwanggyo.sale

- **5. 양주신도시**

- 상권 결과 : yangju.sale

- **6. 동탄신도시**

- 상권 결과 : dongtan.sale

< Code 6 - 기업등록부 상용근로자 >[5)]

▶ 코드 설명 : 신도시별 기업등록부(사업자 기준)에 있는 BR 사업체 상용근로자 합계 및 비율을 보기 위한 코드
▶ Code 1과 연계되므로 Code 1을 실행해야 작동됨.

1) 기업등록부 원데이터 호출

```
> if(!require(data.table)) {
+    install.packages("data.table")
+ }
> library(data.table)

> if(!require(bit64)) {
+    install.packages("bit64")
+ }
> library(bit64)

> company15 <- fread("기업등록부_사업자_2015.csv", stringsAsFactors = F)
> company16 <- fread("기업등록부_사업자_2016.csv", stringsAsFactors = F)
```

2) 1기 신도시 : BR 사업체 상용근로자 합계 및 비율 추출

- 1. 분당신도시

```
> am_bundang <- c(3102362, 3102364, 3102363, 3102360, 3102361, 3102358, 3102359, 3102351, 3102353,
+                 3102354, 3102352, 3102378, 3102377, 3102355, 3102356, 3102371, 3102372)
> bundang.worker15 <- sum(subset(company15$BR_EMP_T, company15$AD_CLS_CD %in% am_bundang))
> bundang.worker16 <- sum(subset(company16$BR_EMP_T, company16$AD_CLS_CD %in% am_bundang))
>bundang.work.tot15 <- sum(subset(company15$BR_EMP_MF_T, company15$AD_CLS_CD %in%
am_bundang))
> bundang.work.tot16 <- sum(subset(company16$BR_EMP_MF_T, company16$AD_CLS_CD %in%
am_bundang))

> bundang.workq15 <- bundang.worker15 / bundang.work.tot15
> bundang.workq16 <- bundang.worker16 / bundang.work.tot16
```

※ 변수명 설명
- am_bundang : 분당신도시 행정구역 코드를 조사한 후 입력한 변수

(1) 상용근로자 합계

```
> bundang.worker15
[1] 93494
> bundang.worker16
[1] 97146
```

(2) 상용근로자 비율

```
> bundang.workq15
[1] 0.6848926
> bundang.workq16
[1] 0.6565871
```

※ 위와 같은 방법으로 총 11개의 1, 2기 신도시 모두에 대하여 상권 코드를 추출하였음.

- 2. 일산신도시

(1) 상용근로자 합계

```
> ilsan.worker15
[1] 55739
> ilsan.worker16
[1] 58813
```

(2) 상용근로자 비율

```
> ilsan.workq15
[1] 0.6969521
> ilsan.workq16
[1] 0.6472356
```

- 3. 평촌신도시

(1) 상용근로자 합계

```
> pyeongchon.worker15
[1] 28226
> pyeongchon.worker16
[1] 28609
```

(2) 상용근로자 비율

```
> pyeongchon.workq15
[1] 0.6838522
> pyeongchon.workq16
[1] 0.6450733
```

- 4. 산본신도시

(1) 상용근로자 합계

```
> sanbon.worker15
[1] 15695
```

> sanbon.worker16
[1] 17079

(2) 상용근로자 비율
> sanbon.workq15
[1] 0.6798787
> sanbon.workq16
[1] 0.6688991

- 5. 중동신도시

(1) 상용근로자 합계
> jungdong.worker15
[1] 59461
> jungdong.worker16
[1] 62825

(2) 상용근로자 비율
> jungdong.workq15
[1] 0.6794689
> jungdong.workq16
[1] 0.6903922

3) 2기 신도시 : BR 사업체 상용근로자 합계 및 비율 추출

- 1. 판교신도시

(1) 상용근로자 합계
> pangyo.worker15
[1] 63160
> pangyo.worker16
[1] 72383

(2) 상용근로자 비율
> pangyo.workq15
[1] 0.7385925
> pangyo.workq16
[1] 0.7290133

- 2. 한강신도시

(1) 상용근로자 합계
> hangang.worker15
[1] 6060
> hangang.worker16
[1] 7191

(2) 상용근로자 비율
> hangang.workq15
[1] 0.6091677
> hangang.workq16
[1] 0.6146679

- 3. 운정신도시

(1) 상용근로자 합계
> unjeong.worker15
[1] 7656
> unjeong.worker16
[1] 8537

(2) 상용근로자 비율
> unjeong.workq15
[1] 0.6284682
> unjeong.workq16
[1] 0.6304557

- **4. 광교신도시**

(1) 상용근로자 합계
```
> gwanggyo.worker15
[1] 27521
> gwanggyo.worker16
[1] 30111
```

(2) 상용근로자 비율
```
> gwanggyo.workq15
[1] 0.6969989
> gwanggyo.workq16
[1] 0.6946341
```

- **5. 양주신도시**

(1) 상용근로자 합계
```
> yangju.worker15
[1] 5526
> yangju.worker16
[1] 5982
```

(2) 상용근로자 비율
```
> yangju.workq15
[1] 0.6959698
> yangju.workq16
[1] 0.6936456
```

- **6. 동탄신도시**

(1) 상용근로자 합계
```
> dongtan.worker15
[1] 21602
> dongtan.worker16
[1] 26198
```

(2) 상용근로자 비율
```
> dongtan.workq15
[1] 0.6352035
> dongtan.workq16
[1] 0.6109038
```

(2) 외부데이터 전처리(R 활용)

< Code 1 - 주택가격 시계열 분석 >

1) KB주택가격동향 - 시계열 전처리

```
> install.packages('xlsx')
> install.packages("rJava")
> Sys.setenv(JAVA_HOME="C:/Program Files/Java/jre1.8.0_211")
> library(xlsx)
```

2) 서울 주택매매지수 파일 분리(종합, 아파트)

- 1. 서울 주택매매지수(종합 : 아파트 + 단독 + 연립)

```
> seoul <- read.xlsx('서울.xlsx',1, encoding = 'UTF-8', stringsAsFactors = F)
> names(seoul) <- c("연도", month.abb)
> write.xlsx(seoul, file = 'seoul.xlsx')
```

- 2. 서울 주택매매지수(아파트)

```
> seoul_A <- read.xlsx('서울.xlsx', 2, encoding = 'UTF-8', stringsAsFactors = F)
> names(seoul_A) <- c("연도", month.abb)
> write.xlsx(seoul_A, file = 'seoul_A.xlsx')
```

< Code 2 - 버스 정류장 크롤링 >

▶ 검색 사이트는 '다음(Daum)'을 이용함.

1) 필요 패키지 설치

```
> if(!require(rvest)) {
+    install.packages('rvest')
+ }
> library(rvest)

> if(!require(dplyr)) {
+    install.packages('dplyr')
+ }
> library(dplyr)
```

2) 다음 검색 크롤링 함수 제작

```
> make_st <- function(url){
+    html <- read_html(url)
+    temp <- html %>% html_nodes('.info') %>% html_nodes('a') %>% html_attr('data-name')
+    temp <- temp[!is.na(temp)]
+    return(temp)
+ }
```

3) 2기 신도시 : 정류장 크롤링

※ 검색 방법 : '경기 ( )시 ( )동 정류장'으로 검색함.

- 1. 판교신도시

(1) 행정동 URL

```
> url_sampyung
<-'https://search.daum.net/search?nil_suggest=btn&w=tot&DA=SBC&q=%EA%B2%BD%EA%B8%B0+%EC%84%B1%EB%82%A8%EC%8B%9C+%EB%B6%84%EB%8B%B9%EA%B5%AC+%EC%82%BC%ED%8F%89%EB%8F%99+%EC%A0%95%EB%A5%98%EC%9E%A5'

> url_beakhyun
<-'https://search.daum.net/search?nil_suggest=btn&w=tot&DA=SBC&q=%EA%B2%BD%EA%B8%B0+%EC%84%B1%EB%82%A8%EC%8B%9C+%EB%B6%84%EB%8B%B9%EA%B5%AC+%EB%B0%B1%ED%98%84%EB%8F%99+%EC%A0%95%EB%A5%98%EC%9E%A5'

> url_pangyo
<-'https://search.daum.net/search?nil_suggest=btn&w=tot&DA=SBC&q=%EA%B2%BD%EA%B8%B0+%EC%84%B1%EB%82%A8%EC%8B%9C+%ED%8C%90%EA%B5%90%EB%8F%99++%EC%A0%95%EB%A5%98%EC%9E%A5'
```

```
> url_unjung
<-'https://search.daum.net/search?nil_suggest=btn&w=tot&DA=SBC&q=%EA%B2%BD%EA%B8%B0+%EC%84%B1%EB%82%A8%EC%8B%9C+%EC%9A%B4%EC%A4%91%EB%8F%99+%EC%A0%95%EB%A5%98%EC%9E%A5'
```

(2) 크롤링 함수 실행해 정보 수집

```
> sampyung_st <- make_st(url_sampyung)
> beakhyun_st <- make_st(url_beakhyun)
> pangyo_st <- make_st(url_pangyo)
> unjung_st <- make_st(url_unjung)
```

(3) 수집한 정보 합치기

```
> Allpangyo_st <- c(sampyung_st, beakhyun_st, pangyo_st, unjung_st)
```

※ 위와 동일한 방법으로 모든 신도시 정류장 정보를 크롤링하였음.[6)]

- **2. 광교신도시**

```
> Allgwanggyo_st <- c(woncheon_st, gwanggyo1_st, gwanggyo2_st, sanghyeon1_st, sanghyeon2_st)
```

- **3. 동탄신도시**

(1) 동탄1신도시, 동탄2신도시 분리하여 조사

```
> Alldongtan1_st <- c(dongtan1, dongtan2, dongtan3)
> Alldongtan2_st <- c(dongtan4, dongtan5, dongtan6)
```

(2) 동탄신도시 총합

```
> Alldongtan_st <- c(Alldongtan1_st, Alldongtan2_st)
```

- **4. 운정신도시**

```
> Allunjeong_st <- c(unjeong1_st, unjeong2_st, unjeong3_st)
```

- **5. 한강신도시**

```
> Allhangang_st <- c(janggi_st, unyang_st, gurae_st, masan_st)
```

- **6. 양주신도시**

```
> Allyangju_st <- c(okjeong_st, hoejeong_st)
```

4) 1기 신도시 : 정류장 크롤링

- **1. 분당신도시**

```
> Allbundang_st <- c(yatap_st, imae_st, seohyeon_st, bundang_st, sunae_st, jeongja_st, gumi1_st)
```

- **2. 일산신도시**

```
> Allilsan_st <- c(janghang2_st, madu_st, baekseok1_st, baekseok2_st, jeongbalsan_st, ilsan1_st, ilsan2_st, ilsan3_st, juyeop_st, deahwa_st)
```

- **3. 평촌신도시**

```
> Allpyeongchon_st <- c(galsan_st, gwiin_st, beomgye_st, burim_st, buheung_st, sinchon_st, pyeongan_st, pyeongchon_st)
```

- **4. 산본신도시**

```
> Allsanbon_st <- c(sanbon1_st, sanbon2_st, jaegung_st, ogeum_st, suri_st, gungnae_st, gwangjeong_st)
```

- **5. 중동신도시**

```
> Alljungdong_st <- c(jungdong_st, jung1_st, jung2_st, jung3_st, jung4_st, simgok_st, songnae_st, naedong_st, samjeong_st)
```

< Code 3 - 버스 정류장 정보 >

▶ 외부데이터 분석 Code 2과 연계되므로 Code 2를 실행해야 작동됨.

1) 정류장 정보 csv file 호출

```
> if(!require(data.table)) {
+    install.packages("data.table")
+ }
```

```
> library(data.table)

> stationIF <- fread('stationIF.csv')
```

2) 정류소 추출 함수 제작

```
> station <- function(Allstation){
+   citystation <- subset(stationIF, stationIF$STATION_NM %in% Allstation)
+   temp <- unique(Allstation) %in% unique(citystation$STATION_NM)
+   if(FALSE %in% temp){
+     cat('미포함정류장:', unique(Allstation)[!temp])
+   }
+   return(citystation)
+ }
```

3) 2기 신도시 : 정류장 추출

- 1. 판교신도시

```
> pangyo_station <- station(Allpangyo_st)
```

- 2. 광교신도시

```
> gwanggyo_station <- station(Allgwanggyo_st)
```

- 3. 동탄신도시

```
> dongtan_station <- station(Alldongtan_st)
```

- 4. 운정신도시

```
> unjeong_station <- station(Allunjeong_st)
```

- 5. 한강신도시

```
> hangang_station <- station(Allhangang_st)
```

- 6. 양주신도시

```
> yangju_station <- station(Allyangju_st)
```

4) 1기 신도시 : 정류장 추출

- 1. 분당신도시

```
> bundang_station <- station(Allbundang_st)
```

- 2. 일산신도시

```
> ilsan_station <- station(Allilsan_st)
```

- 3. 평촌신도시

```
> pyeongchon_station <- station(Allpyeongchon_st)
```

- 4. 산본신도시

```
> sanbon_station <- station(Allsanbon_st)
```

- 5. 중동신도시

```
> jungdong_station <- station(Alljungdong_st)
```

< Code 4 - 신도시별 경유 버스 및 노선번호 >

▶ 외부데이터 분석 Code 3과 연계되므로 Code 3을 실행해야 작동됨.

1) 원데이터 호출

```
> if(!require(dplyr)) {
+   install.packages('dplyr')
+ }
> library(dplyr)
```

- 1. 버스 루트 데이터

```
> RT_1 <- fread('RT_1.csv')
> RT_2 <- fread('RT_2.csv')
```

```
> RT_3 <- fread('RT_3.csv')
```

- 2. 버스 루트 리스트

```
> RT_list <- list()
> RT_list[[1]] <- RT_1
> RT_list[[2]] <- RT_2
> RT_list[[3]] <- RT_3
```

2. 함수 제작 1 : 각 신도시의 정류장을 지나는 버스와 노선번호 추출

※ bus(버스번호), route(노선번호) 하위 리스트에 저장함.
※ 같은 버스번호에 노선번호가 두개(이상) 할당된 경우 뽑아냄.

```
> Bus_Route <- function(station){
+   tp <- NULL
+   for(i in 1:length(RT_list)){
+     list <- RT_list[[i]]
+     temp <- subset(list, list$STATION_NM %in% station$STATION_NM)
+     tp <- bind_rows(tp, temp)
+   }
+   bs <- unique(tp$ROUTE_NM)
+   for(i in 1:length(bs)){
+     a <- subset(tp, tp$ROUTE_NM == bs[i])
+     if(length(unique(a$ROUTE_ID)) != 1)
+       print(bs[i])
+   }
+   result <- list(bus = unique(tp$ROUTE_NM), route = unique(tp$ROUTE_ID))
+   return(result)
+ }
```

3. 함수 제작 2 : 각 신도시를 지나는 버스 루트 추출

※ 노선번호를 통한 RT 데이터 중 신도시 데이터 추출함.

```
> Allcity_RT <- function(city){
+   temp <- city$route
+   result <- NULL
+   for(i in 1:length(temp)){
+     temp1 <- subset(RT_1, RT_1$ROUTE_ID == temp[i])
+     temp2 <- subset(RT_2, RT_2$ROUTE_ID == temp[i])
+     temp3 <- subset(RT_3, RT_3$ROUTE_ID == temp[i])
+     temp4 <- bind_rows(temp1, temp2, temp3)
+     result <- bind_rows(result, temp4)
+   }
+   return(result)
+ }
```

4) 2기 신도시 : 함수 1 결과

- 1. 판교신도시

```
> pangyo <- Bus_Route(pangyo_station)
```

- 2. 광교신도시

```
> gwanggyo <- Bus_Route(gwanggyo_station)
```

- 3. 동탄신도시

```
> dongtan <- Bus_Route(dongtan_station)
```

- 4. 한강신도시

```
> hangang <- Bus_Route(hangang_station)
```

- 5. 운정신도시

```
> unjeong <- Bus_Route(unjeong_station)
```

- 6. 양주신도시

```
> yangju <- Bus_Route(yangju_station)
```

5) 1기 신도시 : 함수 1 결과

- 1. 분당신도시

```
> bundang <- Bus_Route(bundang_station)
```

- 2. 일산신도시

```
> ilsan <- Bus_Route(ilsan_station)
```

- 3. 평촌신도시

```
> pyeongchon <- Bus_Route(pyeongchon_station)
```

- 4. 산본신도시

```
> sanbon <- Bus_Route(sanbon_station)
```

- 5. 중동신도시

```
> jungdong <- Bus_Route(jungdong_station)
```

6) 2기 신도시 : 함수 2 결과

- 1. 판교신도시

```
> RT_pangyo <- Allcity_RT(pangyo)
```

- 2. 광교신도시

```
> RT_gwanggyo <- Allcity_RT(gwanggyo)
```

- 3. 동탄신도시

```
> RT_dongtan <- Allcity_RT(dongtan)
```

- 4. 한강신도시

```
> RT_hangang <- Allcity_RT(hangang)
```

- 5. 운정신도시

```
> RT_unjeong <- Allcity_RT(unjeong)
```

- 6. 양주신도시

```
> RT_yangju <- Allcity_RT(yangju)
```

7) 1기 신도시 : 함수 2 결과

- 1. 분당신도시

```
> RT_bundang <- Allcity_RT(bundang)
```

- 2. 일산신도시

```
> RT_ilsan <- Allcity_RT(ilsan)
```

- 3. 평촌신도시

```
> RT_pyeongchon <- Allcity_RT(pyeongchon)
```

- 4. 산본신도시

```
> RT_sanbon <- Allcity_RT(sanbon)
```

- 5. 중동신도시

```
> RT_jungdong <- Allcity_RT(jungdong)
```

< Code 5 - 신도시별 서울 소재 정류장 >

▶ 코드 설명 : 신도시별 버스 노선의 정류장에서 서울 소재 정류장만 추출하기 위한 코드
▶ 외부데이터 분석 Code 4과 연계되므로 Code 4를 실행해야 작동됨.

1) 원데이터 호출

```
> seoul_station <- fread('seoul bus station.csv', encoding = 'UTF-8')
> str(seoul_station)
```

2) 2기 신도시 : 신도시를 지나는 버스가 경유하는 서울 정류장 인덱스

```
> temp1 <- seoul_station$'정류소명' %in% RT_pangyo$STATION_NM
> temp2 <- seoul_station$'정류소명' %in% RT_gwanggyo$STATION_NM
> temp3 <- seoul_station$'정류소명' %in% RT_dongtan$STATION_NM
> temp4 <- seoul_station$'정류소명' %in% RT_hangang$STATION_NM
> temp5 <- seoul_station$'정류소명' %in% RT_unjeong$STATION_NM
> temp7 <- seoul_station$'정류소명' %in% RT_yangju$STATION_NM
```

3) 2기 신도시 : 서울 정류장 추출

- 1. 판교신도시

```
> pangyo_seoul <- seoul_station[temp1]
```

- 2. 광교신도시

```
> gwanggyo_seoul <- seoul_station[temp2]
```

- 3. 동탄신도시

```
> dongtan_seoul <- seoul_station[temp3]
```

- 4. 한강신도시

```
> hangang_seoul <- seoul_station[temp4]
```

- 5. 운정신도시

```
> unjeong_seoul <- seoul_station[temp5]
```

- 6. 양주신도시

```
> yangju_seoul <- seoul_station[temp7]
```

4) 1기 신도시 : 신도시를 지나는 버스가 경유하는 서울 정류장 인덱스

```
> temp8 <- seoul_station$'정류소명' %in% RT_bundang$STATION_NM
> temp9 <- seoul_station$'정류소명' %in% RT_ilsan$STATION_NM
> temp10 <- seoul_station$'정류소명' %in% RT_pyeongchon$STATION_NM
> temp11 <- seoul_station$'정류소명' %in% RT_sanbon$STATION_NM
> temp12 <- seoul_station$'정류소명' %in% RT_jungdong$STATION_NM
```

5) 1기 신도시 : 서울 정류장 추출

- 1. 분당신도시

```
> bundang_seoul <- seoul_station[temp8]
```

- 2. 일산신도시

```
> ilsan_seoul <- seoul_station[temp9]
```

- 3. 평촌신도시

```
> pyeongchon_seoul <- seoul_station[temp10]
```

- 4. 산본신도시

```
> sanbon_seoul <- seoul_station[temp11]
```

- 5. 중동신도시

```
> jungdong_seoul <- seoul_station[temp12]
```

---

1) 첨부파일 'Code 1 - 소지역 코드 분류' 참조
2) 첨부파일 'Code 2 - 업종별매출' 참조
3) 첨부파일 'Code 3 - 유동인구 총합, 추이 및 남녀 비율' 참조
4) 첨부파일 'Code 5 - 신도시별 상권' 참조
5) 첨부파일 'Code 6 - 기업등록부 상용근로자' 참조
6) 첨부파일 '1. bus station crawling' 참조