R 구문 비교 : : 컨닝 쪽지

달러($) 기호 구문

목표함수명(데이터$x, 데이터$y)

요약 통계량:

연속형 변수 1개:
```
mean(mtcars$mpg)
```

범주형 변수 1개:
```
table(mtcars$cyl)
```

범주형 변수 2개:
```
table(mtcars$cyl, mtcars$am)
```

연속형 1개, 범주형 1개:
```
mean(mtcars$mpg[mtcars$cyl==4])
mean(mtcars$mpg[mtcars$cyl==6])
mean(mtcars$mpg[mtcars$cyl==8])
```

플롯 그래프:

연속형 변수 1개:
```
hist(mtcars$disp)
boxplot(mtcars$disp)
```

범주형 변수 1개:
```
barplot(table(mtcars$cyl))
```

연속형 변수 2개:
```
plot(mtcars$disp, mtcars$mpg)
```

범주형 변수 2개:
```
mosaicplot(table(mtcars$am, mtcars$cyl))
```

연속형 1개, 범주형 1개:
```
histogram(mtcars$disp[mtcars$cyl==4])
histogram(mtcars$disp[mtcars$cyl==6])
histogram(mtcars$disp[mtcars$cyl==8])

boxplot(mtcars$disp[mtcars$cyl==4])
boxplot(mtcars$disp[mtcars$cyl==6])
boxplot(mtcars$disp[mtcars$cyl==8])
```

데이터 정제작업(WRANGLING):

부분집합 뽑아내기(subsetting):
```
mtcars[mtcars$mpg>30, ]
```

신규 변수 생성:
```
mtcars$efficient[mtcars$mpg>30] <- TRUE
mtcars$efficient[mtcars$mpg<30] <- FALSE
```

모형공식 구문

목표함수명(y~x|z, data=데이터, group=w)

요약 통계량:

연속형 변수 1개:
```
mosaic::mean(~mpg, data=mtcars)
```

범주형 변수 1개:
```
mosaic::tally(~cyl, data=mtcars)
```

범주형 변수 2개:
```
mosaic::tally(cyl~am, data=mtcars)
```

연속형 1개, 범주형 1개:
```
mosaic::mean(mpg~cyl, data=mtcars)
```

틸드(tilde)

플롯 그래프:

연속형 변수 1개:
```
lattice::histogram(~disp, data=mtcars)
lattice::bwplot(~disp, data=mtcars)
```

범주형 변수 1개:
```
mosaic::bargraph(~cyl, data=mtcars)
```

연속형 변수 2개:
```
lattice::xyplot(mpg~disp, data=mtcars)
```

범주형 변수 2개:
```
mosaic::bargraph(~am, data=mtcars, group=cyl)
```

연속형 1개, 범주형 1개:
```
lattice::histogram(~disp|cyl, data=mtcars)

lattice::bwplot(cyl~disp, data=mtcars)
```

R 구문 다양성을 통해 동일한 작업을 수행하는 많은 방식이 제공된다.

동일한 문제를 접근하는데 구문이 얼마나 다른지 컨닝 쪽지를 통해 살펴본다.

깔끔한 세상(Tidyverse) 구문

데이터 %>% 목표함수명(x)

요약 통계량:

연속형 변수 1개:
```
mtcars %>% dplyr::summarize(mean(mpg))
```

범주형 변수 1개:
```
mtcars %>% dplyr::group_by(cyl) %>%
dplyr::summarize(n())
```

파이프(pipe)

범주형 변수 2개:
```
mtcars %>% dplyr::group_by(cyl, am) %>%
dplyr::summarize(n())
```

연속형 1개, 범주형 1개:
```
mtcars %>%  dplyr::group_by(cyl) %>%
  dplyr::summarize(mean(mpg))
```

플롯 그래프:

연속형 변수 1개:
```
ggplot2::qplot(x=mpg, data=mtcars, geom = "histogram")
ggplot2::qplot(y=disp, x=1, data=mtcars, geom="boxplot")
```

범주형 변수 1개:
```
ggplot2::qplot(x=cyl, data=mtcars, geom="bar")
```

연속형 변수 2개:
```
ggplot2::qplot(x=disp, y=mpg, data=mtcars, geom="point")
```

범주형 변수 2개:
```
ggplot2::qplot(x=factor(cyl), data=mtcars, geom="bar") +
      facet_grid(.~am)
```

연속형 1개, 범주형 1개:
```
ggplot2::qplot(x=disp, data=mtcars, geom = "histogram") +
      facet_grid(.~cyl)

ggplot2::qplot(y=disp, x=factor(cyl), data=mtcars,
               geom="boxplot")
```

데이터 정제작업(WRANGLING):

부분집합 뽑아내기(subsetting):
```
mtcars %>% dplyr::filter(mpg>30)
```

신규 변수 생성:
```
mtcars <- mtcars %>%
dplyr::mutate(efficient = if_else(mpg>30,  TRUE, FALSE))
```

R 구문 비교 : : 컨닝 쪽지

구문(Syntax)

프로그래밍 언어에서 어떤 코드는 동작하고 어떤 코드는 동작하지 않는지 관할하는 규칙의 집합이다. 대부분의 프로그래밍 언어는 한가지 표준화된 구문을 제공하지만, R 언어는 팩키지 개발자가 자체 구문을 제작하여 명세하는 것이 허락되어 있다. 결과로, 매우 다양한 (동일하게 적법한) R 구문이 존재하게 되었다.

가장 널리 퍼진 R 구문은 다음과 같다:

1. **달러($) 기호 구문,** 때때로 **베이스 R 구문**으로 불리며 대부분의 베이스 R 함수에서 활용된다. **데이터셋$변수명** 형태로 특징되고, **데이터셋[1,2]**와 같이 꺾쇠 괄호 부분집합 추출하는 것과 연관된다. 거의 모든 R 함수는 달러기호 구문을 인자로 전달하는 것이 허용한다.
2. **모형공식 구문**, `lm()`, `lattice` 그래픽 같은 모형함수에서 사용됨. 하나 (혹은 다수) 예측변수를 종속변수에 연결시키는데 틸드(~)를 사용한다. 다수 베이스 R 함수는 모형공식 구분을 받아들인다.
3. **깔끔한 세상(tidyverse syntax),** `dplyr`, `tidyr`, 등에서 사용된다. 이런 유형의 함수는 데이터가 첫번째 인자로 상정하고, `magrittr` 팩키지 "파이프" (%>%) 연산자와 함께 동작할 수 있도록 한다. 일반적으로 `ggplot2`는 깔끔한 세상(`tidyverse`) 생태계 일원으로 간주되지만, 자체 구문으로 더하기 기호(+)를 사용해서 문자열 조각을 결합하는데 사용한다. Hadley Wickham은 파이프를 학습한 후에 `ggplot2`를 작성했다면 지금과는 다른 구문을 갖게되었을 것이라 말을 했다.

교육서비스를 제공하는 입장에서 통일된 하나의 구문을 가르치려고 하지만, 대부분의 R 프로그래머는 다양한 구문을 조합해서 사용하는 것이 현실이다.

인터넷 검색 조언:

구글, 스택오버플로우, 혹은 여타 선호하는 온라인 웹사이트를 검색하고 나서 평소 인지하지 못한 구문으로 작성된 R코드를 보게되면:

- 컨닝 쪽지에 등재된 일반적인 3가지 구문 중 하나를 사용하고 있는지 확인한다.
- 구문명칭("tidyverse")에서 언급된 키워드 혹은 연관된 팩키지("mosaic")를 사용해서 검색을 다시 시도해보세요.

? 때때로 특정 구분은 동작하지만, 이와 같은 방식으로 활용하는 것은 위험천만하다고 간주된다. 왜냐하면 너무나도 쉽게 잘못되기 때문이다. 예를 들어 변수명을 넘기는데 명칭없이 인자로 넘기는 것이 이에 해당된다.

동일한 작업을 기술하는 더 많은 방법

하나의 구문내에서도 조차도, 동일하게 적법하게 동작하는 변종이 존재한다. 일례로, `ggplot2` 구문을 살펴보자. `ggplot2`는 깔끔한 세상(`tidyverse`)를 구성하는 플롯 그래프 팩키지로 유명하다. 다음을 쭉 읽어나가면, 여기 기술된 코드는 모두 동일한 그래프를 산출하게 되는 것을 알게 된다.

quickplot

`qplot()`은 quickplot의 줄임말로, 신속하게 그래프를 작성할 수 있게 해준다. `ggplot2` 전체 기능을 갖추고 못하고, `ggplot2` 나머지와 비교하여 다소 차이나는 구문을 사용하고 있다.

```
ggplot2::qplot(x=disp, y=mpg, data=mtcars, geom="point")

ggplot2::qplot(x=disp, y=mpg, data=mtcars)  ?

ggplot2::qplot(disp, mpg, data=mtcars)  ? ?
```

ggplot

To unlock the power of `ggplot2` 팩키지의 진정한 힘을 깨우려면, `ggplot()` 함수를 사용(플롯 영역을 구성)해서 `geoms` 을 플롯에 추가할 필요가 있다.

```
ggplot2::ggplot(mtcars) +
    geom_point(aes(x=disp, y=mpg))

ggplot2::ggplot(data=mtcars) +
    geom_point(mapping=aes(x=disp, y=mpg))

ggplot2::ggplot(mtcars, aes(x=disp, y=mpg)) +
    geom_point()

ggplot2::ggplot(mtcars, aes(x=disp)) +
    geom_point(aes(y=mpg))
```

더하기 기호는 계층을 추가

ggformula

모형공식 구문을 사용하는 3.5번째 방식으로 ggplot2 스타일 그래프를 얻을 수 있다.

```
ggformula::gf_point(mpg~disp, data= mtcars)
```

베이스 플롯 그래프에 모형공식

흔히 사용되지는 않지만, 베이스 R 플롯은 모형공식 구문도 받아들인다.

```
plot(mpg~disp, data=mtcars)
```

다르게 보이지만, 동일한 플롯 그래프를 생성하는 구문으로 작성된 코드를 쭉 읽어보세요.