

Ch14. 통계 기법을 이용한 가설 검정

14-1. 가설 검정이란?

가설 검정은 주어진 데이터를 사용하여 특정 가설이 맞는지 틀린지를 결정하는 과정

> 통계적 검정: 수집된 데이터를 사용하여 통계적인 검정을 수행하여(t-검정, chi-squared 검정, ANOVA 등)

> 결과 해석: 계산된 통계량을 바탕으로 가설을 기각할지, 기각하지 않을지를 결정

>> 귀무가설(Null Hypothesis, H_0): 연구자가 기본적으로 거부하려고 하는 가설로 일반적으로는 어떠한 차이나 효과가 없다는 가설

>> 대립가설(Alternative Hypothesis, H_1 또는 H_a): 연구자가 입증하고자 하는 가설로 어떠한 차이나 효과가 있다는 가설

> 유의 확률(significant probability, p-value)은 의미 있는 확률(유사할 확률: 우연에 의해 발생할 확률)

>> 차이가 없다고(귀무 가설이 참이라고) 가정했는데, 그 보다 더 극단적인 (더 차이가 없는) 결과가 나타날 확률

>> $p\text{-value} < 0.05$: 귀무가설을 기각하고 대립가설을 채택 (차이 있음, 우연일 확률이 낮음))

>> $p\text{-value} \geq 0.05$: 귀무가설을 기각하지 않고 유의하지 않다고 판단 (거의 차이 없음, 우연에 의해 발생할 확률이 높을 뿐)

14-2. t 검정 - 두 집단의 평균 비교하기

t-검정(t-test)은 두 집단 간 평균 차이가 통계적으로 유의미한지를 판단하기 위해 사용되는 통계적인 방법 중 하나

> 표본 평균의 차이를 표준 오차로 나눈 t-통계량을 계산하고, 이 t-통계량을 특정 분포(일반적으로 t-분포)와 비교하여 유의성을 판단

compact 자동차와 suv 자동차의 도시 연비 t 검정

```
In [2]: ## 미국 자동차 모델별 연비 자료 데이터 구축
import pandas as pd
mpg = pd.read_csv('mpg.csv')
```

```
In [8]: ## compact와 suv의 도시 연비에 대한 t-검증
# [compact, suv] category별, 빈도 n, cty 평균 구하기
mpg.query('category in ["compact", "suv"]') \
    .groupby('category', as_index = False) \
    .agg(n = ('category', 'count'),
         mean = ('cty', 'mean'))
```

```
Out[8]:    category   n      mean
          0 compact  47  20.12766
          1       suv  62  13.50000
```

```
In [9]: ## t-검증(t-test) 대상 자료 추출
compact = mpg.query('category == "compact")['cty']
suv = mpg.query('category == "suv")['cty']
```

```
In [10]: ## t-검증(t-test)
from scipy import stats
stats.ttest_ind(compact, suv, equal_var = True) #두 집단의 분산(var)이 같다고 가정
Out[10]: Ttest_IndResult(statistic=11.917282584324107, pvalue=2.3909550904711282e-21)
```

: p-value < 0.05이므로 두 집단간 cty의 연비 차이가 통계적으로 의미가 있다고 결론 내릴 수 있음.

일반 휘발유와 고급 휘발유의 도시 연비 t 검정

```
In [11]: ## 휘발류(fl=r)와 고급 휘발류(fl=p)의 도시 연비에 대한 t-검증
# [r, p] fl별 빈도 n, cty 평균 구하기
mpg.query('fl in ["r", "p"]') \
    .groupby('fl', as_index = False) \
    .agg(n = ('category', 'count'),
         mean = ('cty', 'mean'))
```

```
Out[11]:    fl   n      mean
          0   p   52  17.365385
          1   r  168  16.738095
```

```
In [12]: ## t-검증(t-test) 대상 자료 추출
regular = mpg.query('fl == "r")['cty']
premium = mpg.query('fl == "p")['cty']
```

```
In [13]: ## t-검증(t-test)
stats.ttest_ind(regular, premium, equal_var = True)
```

```
Out[13]: Ttest_IndResult(statistic=-1.066182514588919, pvalue=0.28752051088667036)
```

: p-value > 0.05이므로 두 집단간 cty의 연비 차이가 통계적으로 의미가 없다고 결론 내릴 수 있음.

```
In [ ]:
```

14-3. 상관분석 - 두 변수의 관계 분석하기

상관 분석(correlation analysis)은 두 변수 간에 어떠한 선형적인 관계가 있는지, 그 관계의 강도와 방향은 무엇인지 등을 알아보는 데 사용

상관 분석에서 주로 사용되는 지표는 "상관 계수(correlation coefficient)"

- > 1에 가까운 값: 양의 상관 관계. 즉, 한 변수가 증가하면 다른 변수도 증가
- >-1에 가까운 값: 음의 상관 관계. 한 변수가 증가하면 다른 변수는 감소
- > 0에 가까운 값: 상관 관계가 거의 없거나 매우 약한 경우

실업자 수와 개인 소비 지출의 상관관계

1. 상관계수 구하기

```
In [14]: ## economics 데이터 불러오기
economics = pd.read_csv('economics.csv')
```

```
In [15]: ## 상관행렬 만들기: 실업자 수와 개인 소비 지출 사이의 상관 관계
economics[['unemploy', 'pce']].corr()
```

```
Out[15]:
```

	unemploy	pce
unemploy	1.000000	0.614518
pce	0.614518	1.000000

: corr = 0.61 이므로 실업자 수와 개인 소비 지출 사이의 상관은 약 61% 확률로 관계가 있음

2. 유의확률 구하기

```
In [ ]: ## 상관분석: 상관 계수와 유의 확률 구하기
stats.pearsonr(economics['unemploy'], economics['pce'])
```

```
Out[ ]: PearsonRResult(statistic=0.614517614193208, pvalue=6.773527303291701e-61)
```

: corr = 0.61 이므로 실업자 수와 개인 소비 지출 사이의 상관은 약 61% 확률로 관계가 있음

: p-value < 0.05 이므로 두 집단간 차이가 통계적으로 의미가 있다고 결론 내릴 수 있음(우연일 확률이 낮음)

```
In [ ]:
```

상관행렬 히트맵 만들기

1. 상관행렬 만들기

```
In [18]: ## mtcars 데이터 구축하기: 미국 자동차 32종에 대한 11개 종류의 데이터 제공
mtcars = pd.read_csv('mtcars.csv')
mtcars.head()
```

Out[18]:

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

In [19]:

```
## 상관행렬 만들기
car_cor = mtcars.corr()          # 상관행렬 만들기
car_cor = round(car_cor, 2)       # 소수점 둘째 자리까지 반올림
car_cor
```

Out[19]:

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
mpg	1.00	-0.85	-0.85	-0.78	0.68	-0.87	0.42	0.66	0.60	0.48	-0.55
cyl	-0.85	1.00	0.90	0.83	-0.70	0.78	-0.59	-0.81	-0.52	-0.49	0.53
disp	-0.85	0.90	1.00	0.79	-0.71	0.89	-0.43	-0.71	-0.59	-0.56	0.39
hp	-0.78	0.83	0.79	1.00	-0.45	0.66	-0.71	-0.72	-0.24	-0.13	0.75
drat	0.68	-0.70	-0.71	-0.45	1.00	-0.71	0.09	0.44	0.71	0.70	-0.09
wt	-0.87	0.78	0.89	0.66	-0.71	1.00	-0.17	-0.55	-0.69	-0.58	0.43
qsec	0.42	-0.59	-0.43	-0.71	0.09	-0.17	1.00	0.74	-0.23	-0.21	-0.66
vs	0.66	-0.81	-0.71	-0.72	0.44	-0.55	0.74	1.00	0.17	0.21	-0.57
am	0.60	-0.52	-0.59	-0.24	0.71	-0.69	-0.23	0.17	1.00	0.79	0.06
gear	0.48	-0.49	-0.56	-0.13	0.70	-0.58	-0.21	0.21	0.79	1.00	0.27
carb	-0.55	0.53	0.39	0.75	-0.09	0.43	-0.66	-0.57	0.06	0.27	1.00

2. 히트맵 만들기

히트맵 그리기

seaborn.heatmap() 이용

> **data** (필수): 열 지도로 표현할 데이터를 포함하는 2D 배열 또는 Pandas DataFrame을 지정

> **annot**: 각 셀에 값을 표시할지 여부를 설정(기본값은 False)

> **fmt**: annot=True일 때 셀 내의 숫자 값의 서식을 지정

> **cmap**: 열 지도의 색상 맵을 지정('viridis', 'coolwarm', 'Blues', 'RdBu_r' 등의 내장 색상 맵을 사용)

> **cbar**: 색상 막대를 표시할지 여부를 설정(기본값은 True)

> **cbar_kws**: 색상 막대의 추가 매개변수를 설정

> **xticklabels** 및 **yticklabels**: 열 지도의 x축 및 y축 눈금 레이블을 표시할지 여부를 설정

> **square**: 셀이 정사각형인지 여부를 설정

> **linewidths** 및 **linecolor**: 셀 사이의 구분선의 두께와 색상을 설정

> **center**: 색상 스케일의 중심을 조절

> **vmin** 및 **vmax**: 열 지도의 색상 스케일 범위를 지정

> **annot_kws**: `annot=True`일 때 표시된 값의 추가 매개변수를 설정

> **mask**: 특정 값 또는 조건에 따라 특정 셀을 숨기는 데 사용

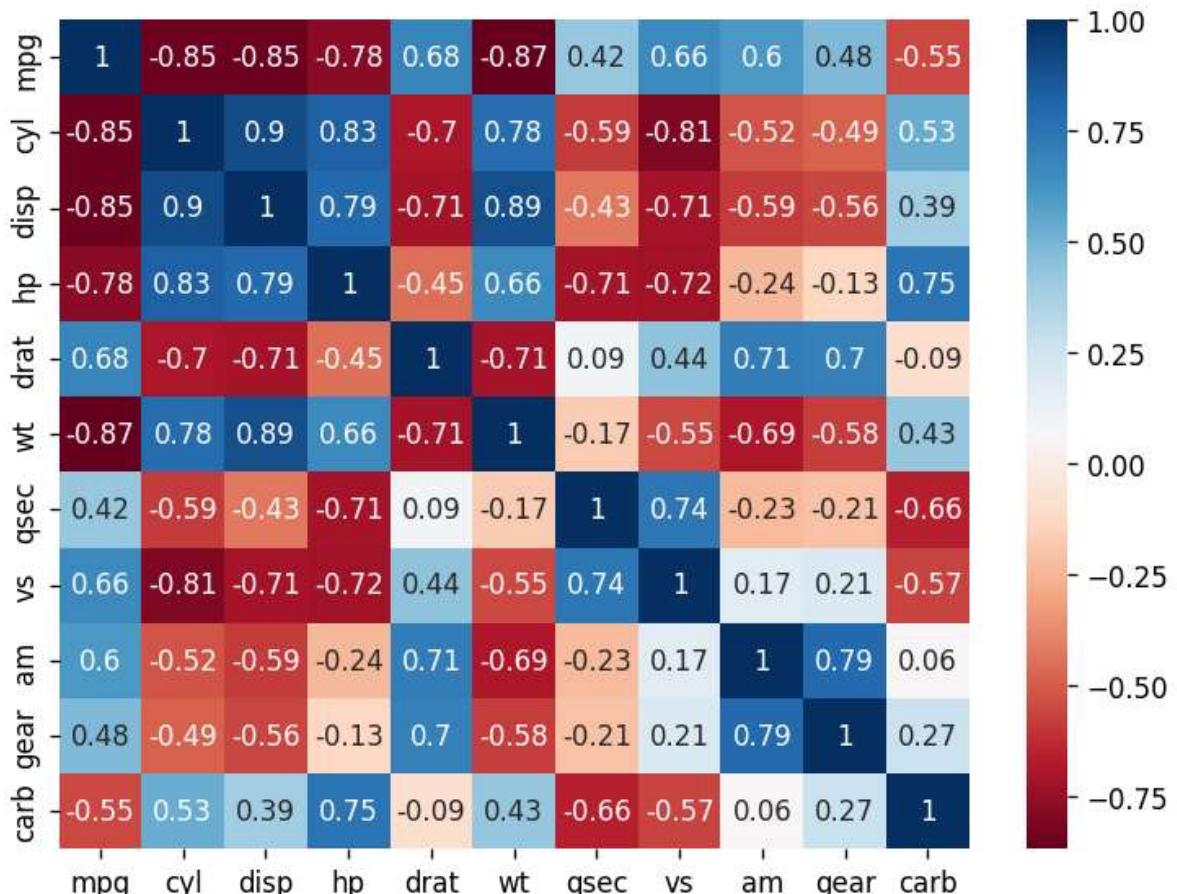
> **ax**: 그래프를 그릴 **Matplotlib** 축을 지정

> **square**: `True`로 설정하면 셀이 정사각형으로 표시

```
In [22]: import matplotlib.pyplot as plt
plt.rcParams.update({'figure.dpi' : '120',           # 해상도 설정
                     'figure.figsize': [7.5, 5.5]}) # 가로 세로 크기 설정
```

```
In [23]: ## 히트맵 만들기
import seaborn as sns
sns.heatmap(car_cor,
            annot = True,    # 상관계수 표시
            cmap = 'RdBu') # 컬러맵
```

Out[23]: <Axes: >



3. 대각 행렬 제거하기

(1) mask 만들기

```
In [25]: ## mask 행렬 만들기
import numpy as np
mask = np.zeros_like(car_cor) # car_cor 크기로 0이 채워진 행렬 생성
mask
```

```
Out[25]: array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
   [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
   [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
   [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
   [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
   [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
   [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
   [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
   [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
   [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
   [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
   [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]])
```

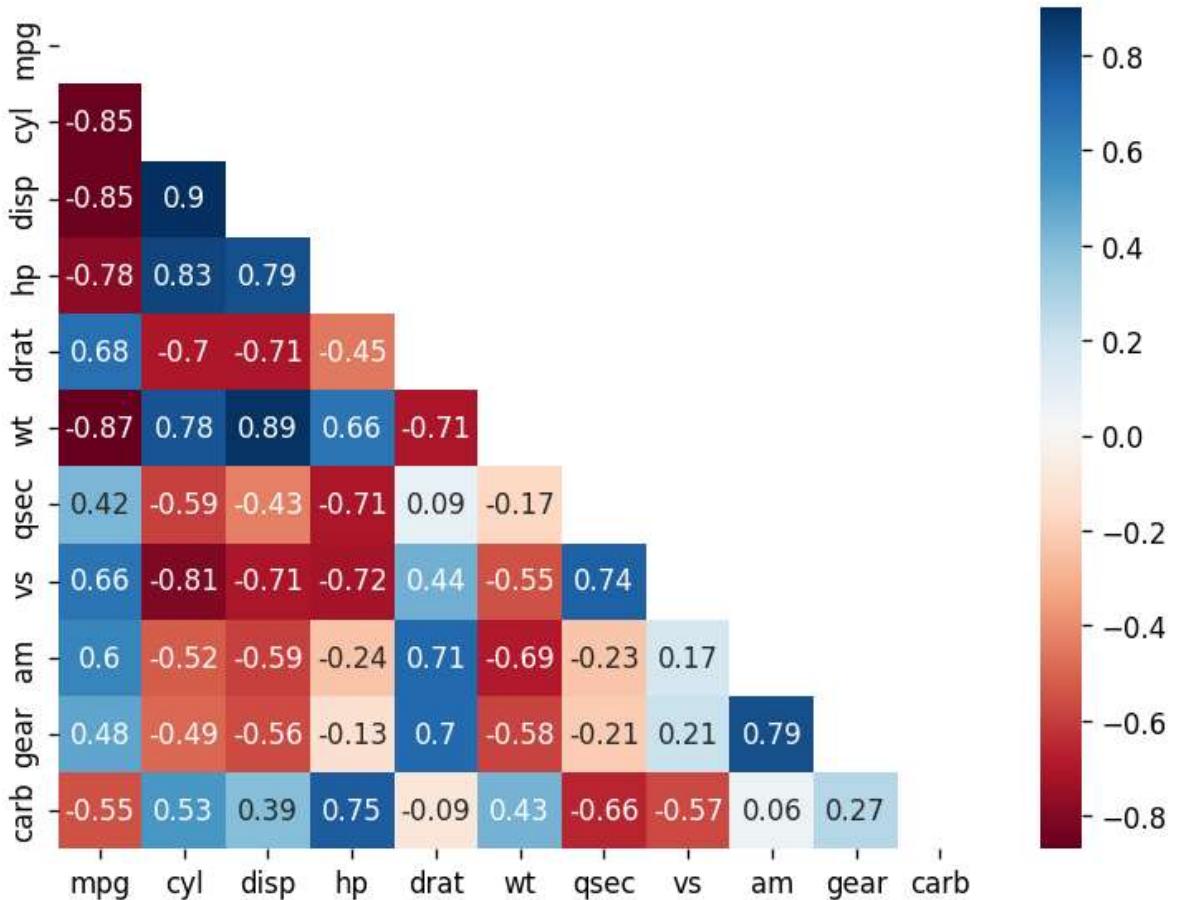
```
In [27]: ## 오른쪽 위 대각 행렬을 1로 바꾸기
mask[np.triu_indices_from(mask)] = 1
mask
```

```
Out[27]: array([[1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
   [0., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
   [0., 0., 1., 1., 1., 1., 1., 1., 1., 1.],
   [0., 0., 0., 1., 1., 1., 1., 1., 1., 1.],
   [0., 0., 0., 0., 1., 1., 1., 1., 1., 1.],
   [0., 0., 0., 0., 0., 1., 1., 1., 1., 1.],
   [0., 0., 0., 0., 0., 0., 1., 1., 1., 1.],
   [0., 0., 0., 0., 0., 0., 0., 1., 1., 1.],
   [0., 0., 0., 0., 0., 0., 0., 0., 1., 1.],
   [0., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
   [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]])
```

(2) 히트맵에 mask 적용하기

```
In [28]: ## 히트맵 만들기: 마스크 행렬 적용
sns.heatmap(data = car_cor,
             annot = True, # 상관계수 표시
             cmap = 'RdBu', # 컬러맵
             mask = mask) # mask 적용
```

```
Out[28]: <Axes: >
```

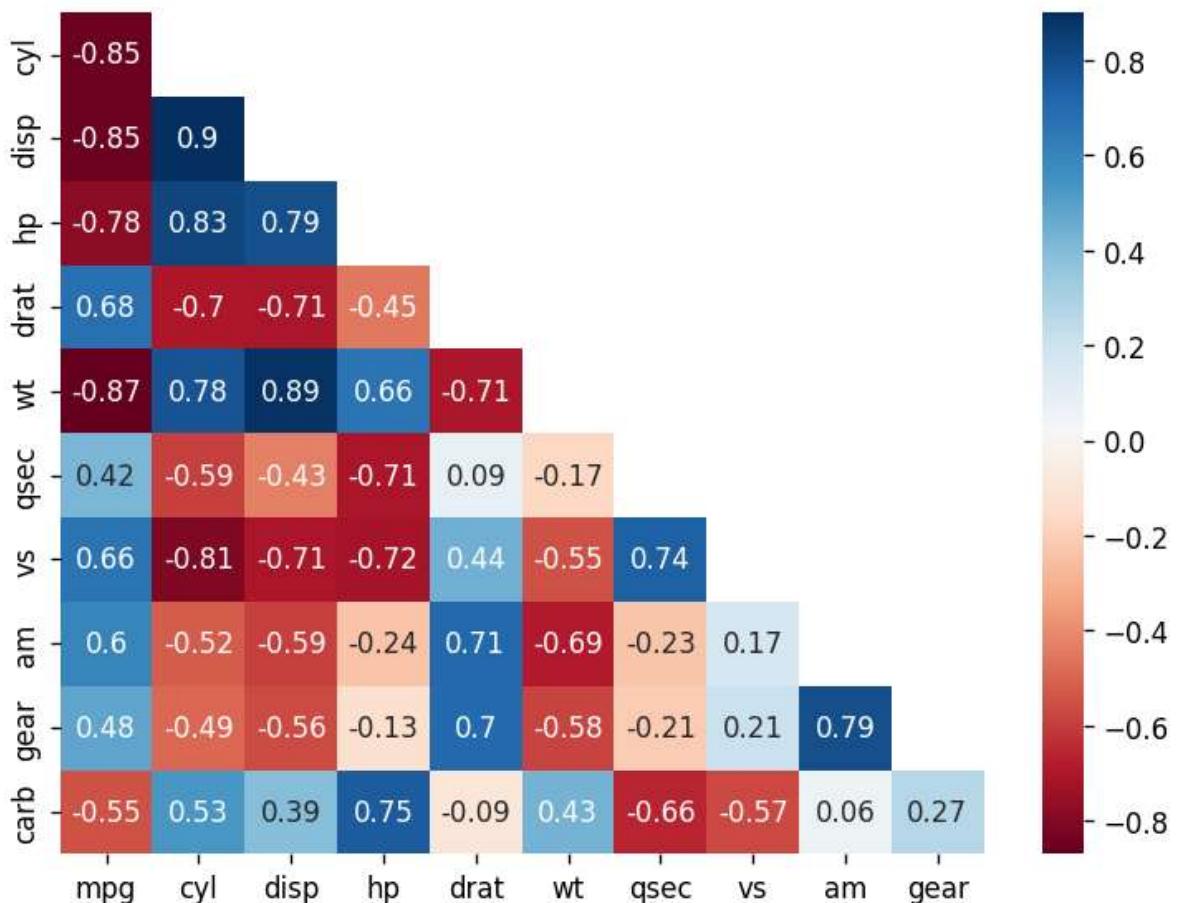


(3) 빈 행과 열 제거하기

```
In [29]: ## 상관계 = 1인 구간 제거하기
mask_new = mask[1:, :-1]          # mask 첫 번째 행, 마지막 열 제거
cor_new = car_cor.iloc[1:, :-1]    # 상관행렬 첫 번째 행, 마지막 열 제거

# 히트맵 만들기
sns.heatmap(data = cor_new,
            annot = True,           # 상관계수 표시
            cmap = 'RdBu',          # 컬러맵
            mask = mask_new)        # mask 적용
```

Out[29]: <Axes: >



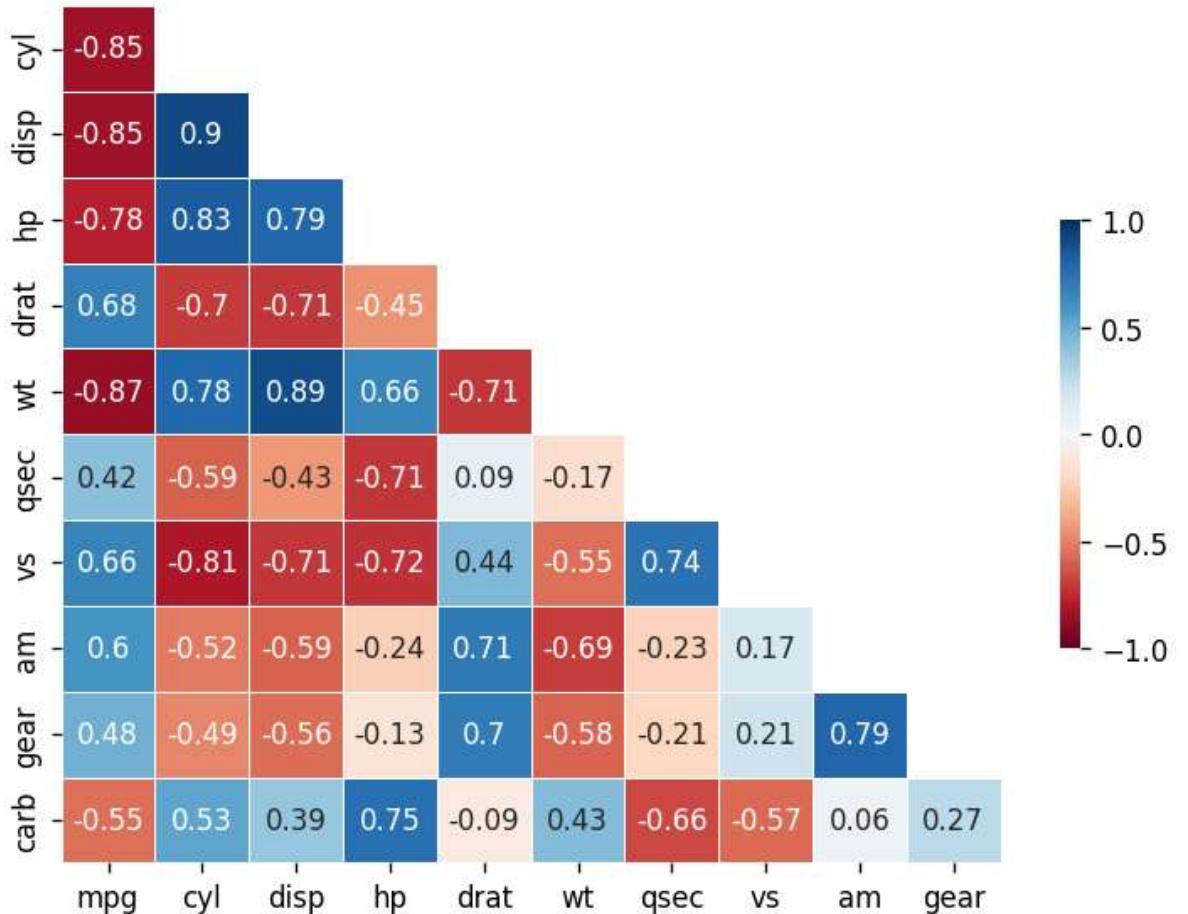
In [30]: `## Mask_new 행렬이 적용된 히트맵 그리기`

```

sns.heatmap(data = cor_new,
             annot = True,                      # 상관계수 표시
             cmap = 'RdBu',                      # 컬러맵
             mask = mask_new,                    # mask 적용
             linewidths = .5,                   # 경계 구분선 추가
             vmax = 1,                          # 가장 진한 파란색으로 표현할 최대값
             vmin = -1,                         # 가장 진한 빨간색으로 표현할 최소값
             cbar_kws = {'shrink': .5})        # 범례 크기 줄이기

```

Out[30]: <Axes: >



In []: