오늘은 PCA(Principal Component Analysis, 주성분 분석)를 알아보겠습니다.

오늘은 PCA 이론적 부분만 볼 예정입니다. 마치 피카츄가 전기를 내뿜을 수 있는 원리를 탐구하는 것과 마찬가지이기 때문에 사진을 넣어봤습니다. 이론적인 내용이라 졸릴 때마다 얘라도 보시면 좋겠습니다. 피카~ 소리가 들리는 것만 같네요.

PCA는 대표적인 차원 축소 기술로 통계 분석뿐만 아니라 영상 분석 등 다차원을 저차원으로 바꿔 새로운 통찰력을 얻을 수 있기 때문에 다양한 곳에서 쓰이는 기법입니다.

쉬운 내용이 아님에도 PCA를 첫 번째 주제로 선택한 이유는, 시각화에 초점을 맞춘 tableau 에게 아주 어울리는 분석 방법이기 때문입니다.
그래서 그런지 태블로사에서도 PCA를 활용한 대시보드를 tabpy 예시로 제공해주고 있습니다.

PCA 내용을 먼저 살펴본 뒤 파이썬으로 구현하고, 태블로에서 실행시켜 결과를 보도록 하겠습니다.

목차

1. PCA 소개

빅데이터 시대를 맞았지만 사람은 3차원까지밖에 시각화하지 못하고 보통 2차원 그래프로 자료를 파악할 수밖에 없습니다.

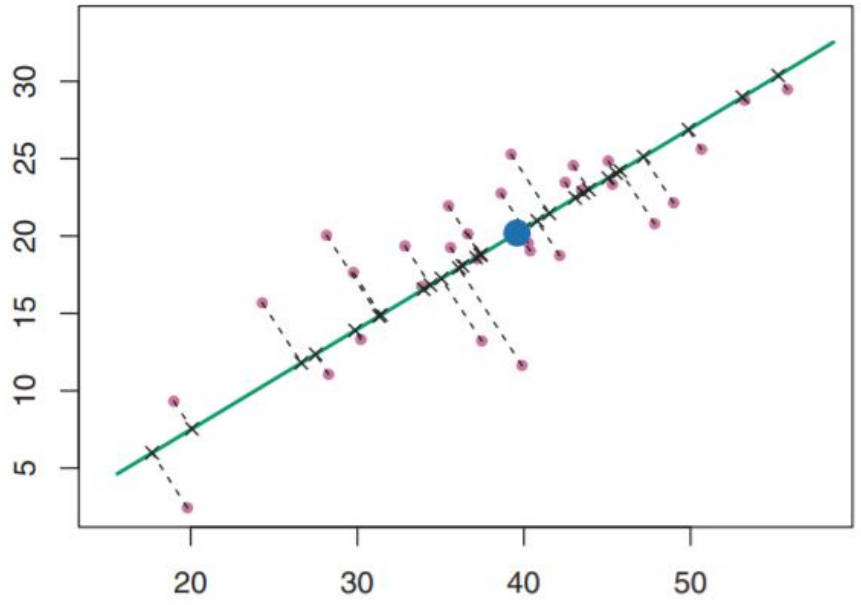
p개 features(columns) 가 있을 때 이를 산점도로 그리면 $p(p-1)/2$ 개의 산점도가 나옵니다.
10개 feature라면 45개의 산점도가 나오네요.

이런 pair는 일일이 다 볼 수도 없고, 다 본다고 해도 분석이 힘듭니다.
모든 feature가 informative 하지도 않습니다. 어떤 것들은 전체 정보를 매우 조금만 담고 있기도 할 거고요.

그러니까 저차원 표현(low-dimensional representation)으로 바꿀 수 있으면 좋겠습니다.
대신 기존 데이터의 정보를 최대한 많이 보존하면서요.

1. PCA 설명

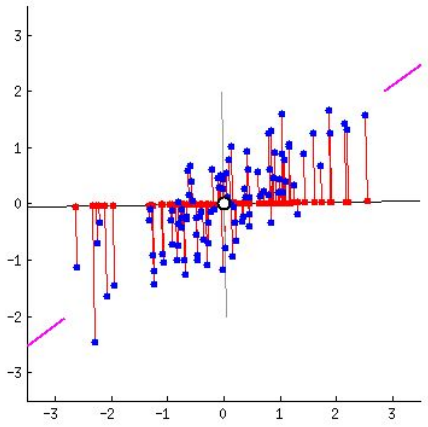
그렇게 다차원 자료를 우리가 파악하기 쉬운 2차원 그림으로 바꾸는 게 목표입니다.
이해하기 쉽게 2차원 데이터의 Principal Component를 묘사한 그림을 먼저 보겠습니다.



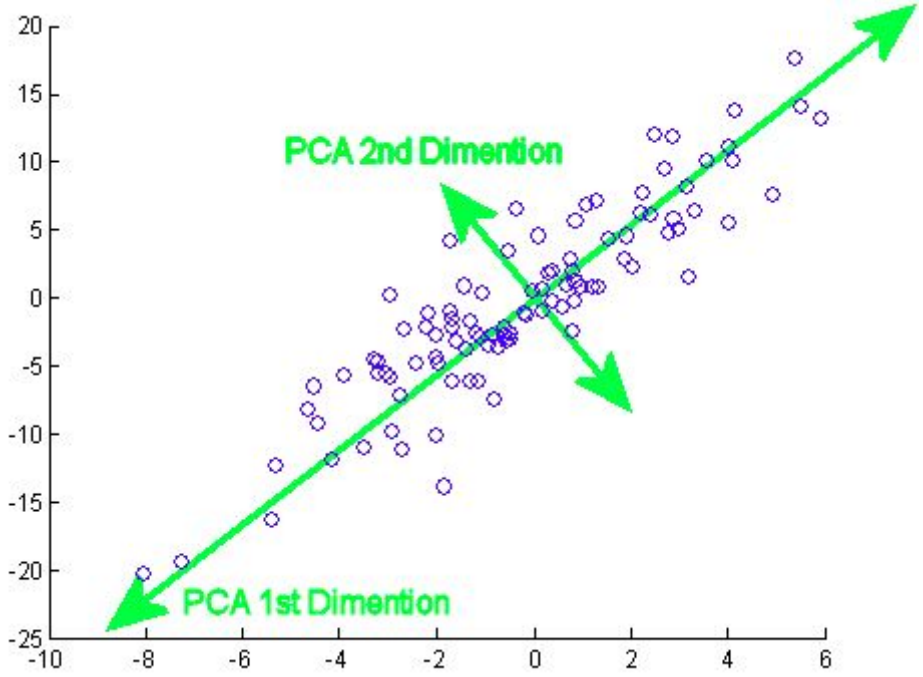
2차원 산점도에 초록색 선을 긋고 점들을 project 하고 있습니다.
이 선의 특징은

1. project 된 결과가 분산(variance)이 가장 크게 저차원으로 만든다.
2. 분산을 최대한 보존한다.
3. 가장 멀리 퍼뜨리는 저차원 표현을 찾는다.
4. 모든 점으로부터 거리가 가장 가까운 라인을 찾는다.
5. 데이터의 정보를 최대한 보존한다.

다 그 말이 그말이쥬?
위 그림은 PC1(Principal Component 1), 첫 번째 주성분을 나타낸 것입니다.
그럼 아닌 경우는 뭔지 아셔야 더 이해도를 높일 수 있겠쥬.



그림과 같이 무수히 많은 라인을 그을 수 있습니다.
project 된 빨간 점의 분포에 초점을 맞춰서 보시기 바랍니다.
빨간 점이 퍼졌다가 모였다가 하는 걸 보실 수 있죠?
그 중 가장 많이 퍼진 라인을 찾겠다는 겁니다.
그 말이 분산이 가장 크고, 가장 멀리 퍼뜨리고, 모든 점으로부터 가장 가깝고, 데이터의 정보를 최대한 보존한다고 보는 거죠.



샘플 수: n
feature 수: p

$$Z_1 = \varphi_{11}X_1 + \varphi_{21}X_2 + \ldots + \varphi_{p1}X_p$$
$$Z_2 = \varphi_{12}X_1 + \varphi_{22}X_2 + \ldots + \varphi_{p2}X_p$$

이렇게 새로운 변수 조합을 계산해 냅니다. 기존 feature X의 선형 조합으로요.

Z_1이 PC1(the 1st principal component) 입니다.

각 feature들을 평균0, 분산 1로 normalize 한 상태에서 계산합니다.(이를 scaling이라고 합니다. 연 1회 건강보험 적용 24,700원 맞나요?)

여러 조합 중 Z_i의 분산이 가장 큰 조합이 계산됩니다.

$\varphi_1 = (\varphi_{11}, \varphi_{21}, \ldots, \varphi_{p1})^T$를 loading vector라 부릅니다. 위의 그래프에서 PC1 라인이 그려진 것이 로딩벡터가 그려진 선입니다.

계수인 $\varphi_{11}, \ldots, \varphi_{p1}$ 은 제곱합 1이 되게 제한합니다. loading vector가 커질수록 Z의 분산도 커지기 때문입니다.

관측치마다 PC1 값이 나오는데요. 아래 식입니다.

$$z_{11} = \varphi_{11}x_{11} + \varphi_{21}x_{12} + \ldots + \varphi_{p1}x_{1p}$$
$$z_{21} = \varphi_{11}x_{21} + \varphi_{21}x_{22} + \ldots + \varphi_{p1}x_{2p}$$
$$\vdots$$
$$z_{n1} = \varphi_{11}x_{n1} + \varphi_{21}x_{n2} + \ldots + \varphi_{p1}x_{np}$$

위 식의 z는 소문자입니다. z_{i1} 은 i 번째 관측치의 PC1 값입니다.
이 z_{i1} 들을 scores of the first principal component 라 부릅니다. 스코어라고 하겠습니다.
계산은 알아서 해주니 한시름 덜었쥬?

PC1이 정해지면 PC2를 구할 수 있는데요.
PC2도 마찬가지 분산이 최대가 되는 선형조합이지만
PC1과 orthogonal,
그러니까 직교,
그러니까 독립적,
그러니까 상관성이 없는 조합이어야 합니다.

$$z_{12} = \varphi_{12}x_{11} + \varphi_{22}x_{22} + \ldots + \varphi_{p2}x_{2p}$$
$$z_{22} = \varphi_{12}x_{21} + \varphi_{22}x_{22} + \ldots + \varphi_{p2}x_{2p}$$
$$\vdots$$

$z_{n2} = \varphi_{12}x_{n2} + \varphi_{22}x_{n2} + \ldots + \varphi_{p2}x_{np}$

PC2의 스코어는 이렇게 되겠네요. PC3, PC4, …, min(n-1,p)까지 구합니다.

위 그림에서는 2차원이기 때문에 직교하는 PC2가 유일하게 정해지네요.
하지만 p가 늘어날수록,
그러니까 feature가 늘어날수록,
그러니까 column이 늘어날수록 벡터의 차원이 커지니 여러 Principal Component 2 후보들이 나올 겁니다.
그 중 maximum variance를 갖는 아해가 PC2가 되겠네요.

3. 예시1 - USArrests

예시 몇 개를 보겠습니다. 불법으로 퍼왔으니 조용히 보시기 바랍니다.

먼저 R에 기본으로 제공되는 USArrests 데이터셋입니다.

	UrbanPop	Murder	Assault	Rape
California	91	9.0	276	40.6
Nevada	81	12.2	252	46.0
West Virginia	39	5.7	81	9.3
Alaska	48	10.0	263	44.5
North Dakota	44	0.8	45	7.3
Maine	51	2.1	83	7.8
Colorado	78	7.9	204	38.7
Virginia	63	8.5	156	20.7
Wyoming	60	6.8	161	15.6
Iowa	57	2.2	56	11.3
South Carolina	48	14.4	279	22.5
New Mexico	70	11.4	285	32.1

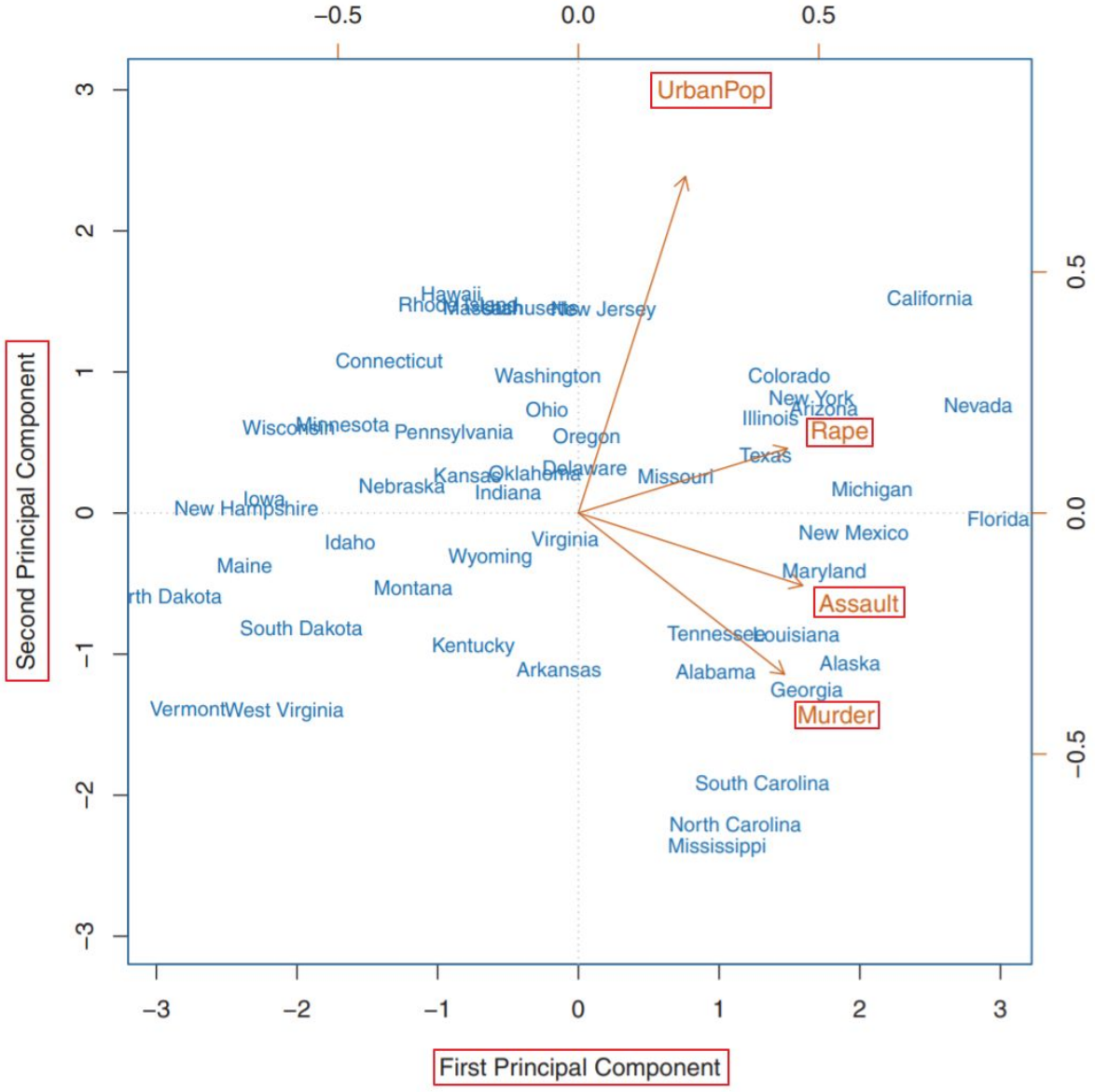
50개 각 주별로 도시거주인구비율, 10만명당 살인, 폭행, 강간 범죄 비율을 나타낸 자료입니다. n=50, p=4 인 자료군요.

	PC1	PC2	PC3	PC4
Murder	-0.536	0.418	-0.341	0.649
Assault	-0.583	0.188	-0.268	-0.743
UrbanPop	-0.278	-0.873	-0.378	0.134
Rape	-0.543	-0.167	0.818	0.089

PC1, PC2, PC3, PC4의 로딩벡터들입니다.
주성분을 보통 min(50-1, 4)개 구하니, 4개의 PC들이 나왔습니다.
그림을 그릴 PC1, PC2를 주목하세요.

	PC1	PC2	PC3	PC4
Alabama	-0.97566045	1.12200121	-0.43980366	0.154696581
Alaska	-1.93053788	1.06242692	2.01950027	-0.434175454
Arizona	-1.74544285	-0.73845954	0.05423025	-0.826264240
Arkansas	0.13999894	1.10854226	0.11342217	-0.180973554
California	-2.49861285	-1.52742672	0.59254100	-0.338559240
Colorado	-1.49934074	-0.97762966	1.08400162	0.001450164
Connecticut	1.34499236	-1.07798362	-0.63679250	-0.117278736
Delaware	-0.04722981	-0.32208890	-0.71141032	-0.873113315
Florida	-2.98275967	0.03883425	-0.57103206	-0.095317042
Georgia	-1.62280742	1.26608838	-0.33901818	1.065974459
Hawaii	0.90348448	-1.55467609	0.05027151	0.893733198
Idaho	1.62331903	0.20885253	0.25719021	-0.494087852
Illinois	-1.36505197	-0.67498834	-0.67068647	-0.120794916
Indiana	0.50038122	-0.15003926	0.22576277	0.420397595
Iowa	2.23099579	-0.10300828	0.16291036	0.017379470
Kansas	0.78887206	-0.26744941	0.02529648	0.204421034
Kentucky	0.74331256	0.94880748	-0.02808429	0.663817237
Louisiana	-1.54909076	0.86230011	-0.77560598	0.450157791
Maine	2.37274014	0.37260865	-0.06502225	-0.327138529
Maryland	-1.74564663	0.42335704	-0.15566968	-0.553450589
Massachusetts	0.48128007	-1.45967706	-0.60337172	-0.177793902

각 관측치(지금은 미국의 주)마다 스코어들이 나왔습니다.
값을 눈여겨보실 필욘 없습니다. 그래프로 나타낼 거니까요.



가로축: PC1, 세로축: PC2
각 주별로 PC1, PC2 스코어들을 나타낸 그래프입니다.
거기에 각 feature별 PC1, PC2 loading vector까지 표현돼 있습니다.
이 그래프를 biplot 이라고 합니다. score와 loading vector가 다 그려져 있기 때문입니다.
여러 가지로 해석할 수 있습니다.

1. PC1 loading vector는 인구보다 범죄들에서 절대값 수치가 높습니다. 따라서 전체적인 중범죄율이라고 해석할 수 있겠네요.

2. PC2 loading vector는 인구가 월등히 높습니다. 그래서 도시화 수준이라고 해석합니다.

3. 세 가지 범죄는 모여있고 인구는 멀리 떨어져 있는 걸 보니 범죄간 양의 상관관계가 있다는 걸 가리키고 있습니다. 반대로 인구는 나머지와 상관관계가 작습니다.

4. 오른쪽에 위치한 California, Nevada, Florida 같은 주들은 PC1에서 높은 양의 값을 갖고 있고 따라서 중범죄율이 높은 곳이라는 걸 알 수 있습니다.

5. 반면 North Dakota 같은 곳들은 PC1이 매우 낮은 음의 값으로 나오고 중범죄율이 낮다는 걸 알 수 있네요.

6. California는 y축, PC2 값 또한 높습니다. 도시화 수준이 높네요.

7. 반면 Mississippi는 반대입니다.

8. 원점에 가까운 Indiana, Virginia 같은 곳들은 도시화나 중범죄율이 다 평균적인 레벨이라고 볼 수 있겠습니다.

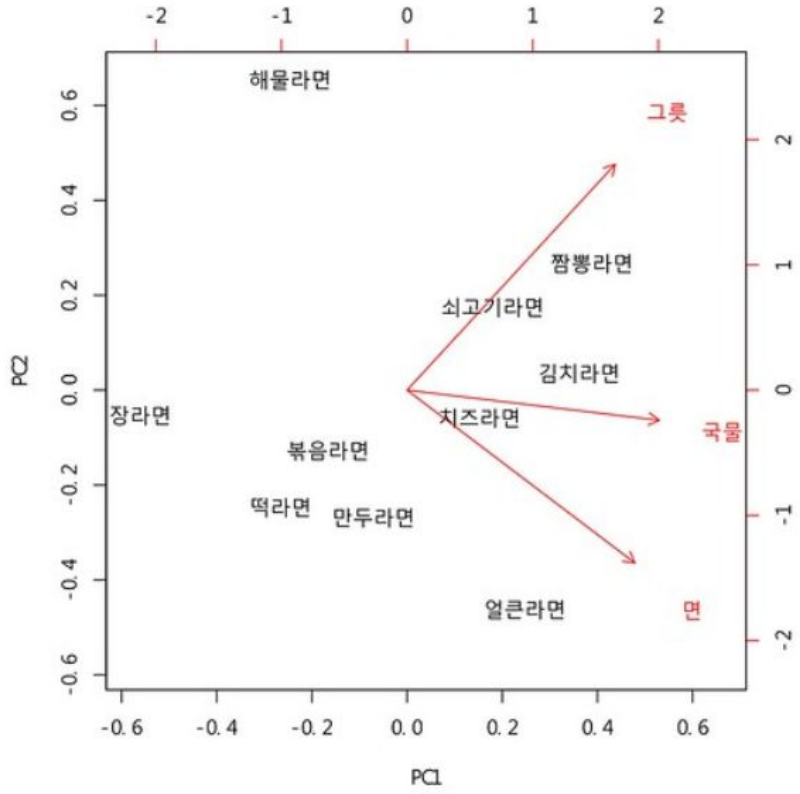
3-ㄴ. 예시2 - 라면

자료가 자료다보니 다소 와닿지 않을 수도 있겠습니다.

누군가 라면으로 재밌는 예제를 만들어 놨는데요. 또다른 친근한 자료도 한 번 보시겠습니다.

	면	그릇	국물
쇠고기라면	2	4	5
해물라면	1	5	1
얼큰라면	5	3	4
떡라면	2	2	3
짬뽕라면	3	5	5
만두라면	4	3	2
치즈라면	4	4	3
된장라면	1	2	1
볶음라면	3	3	2
김치라면	5	5	3

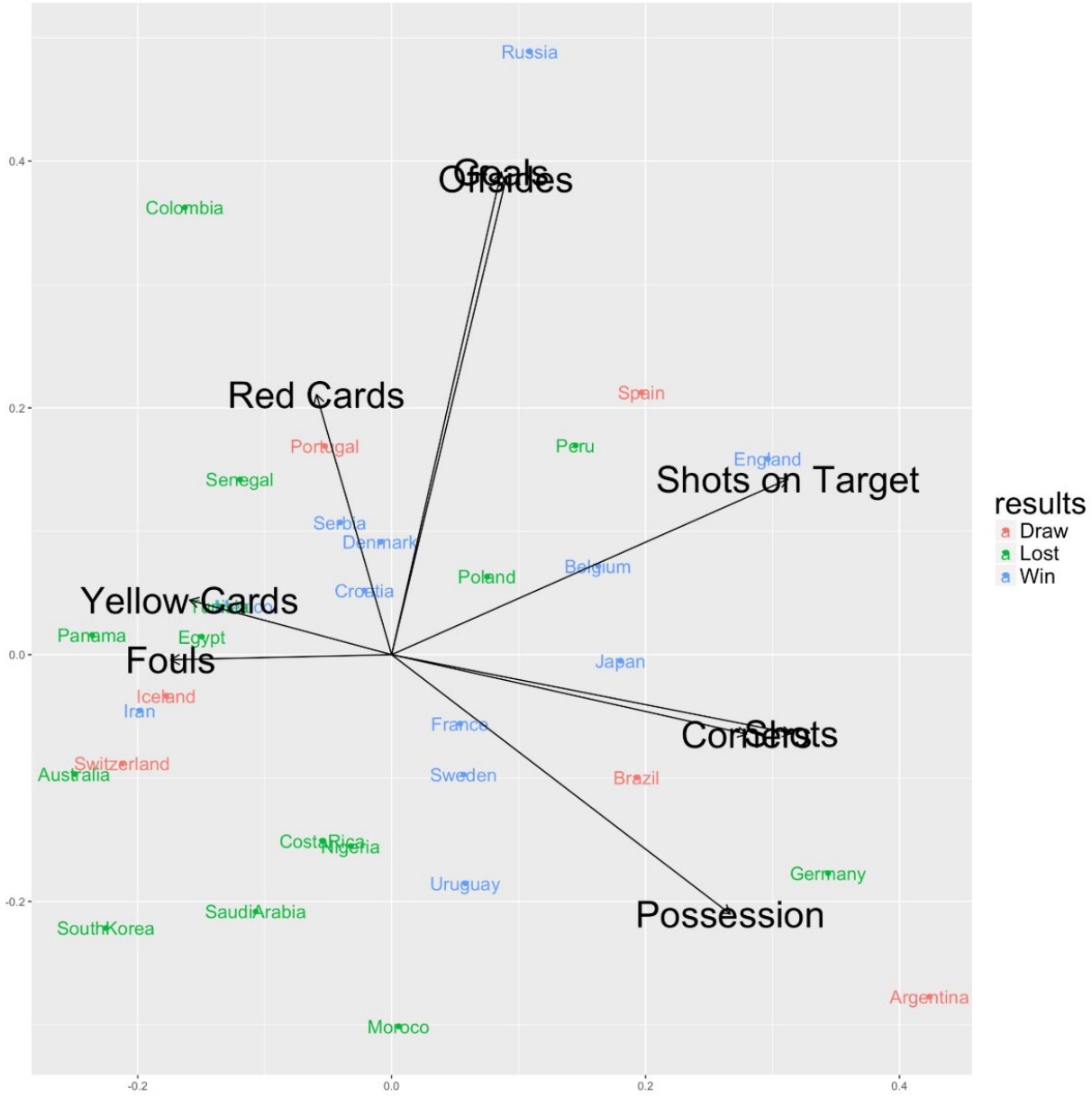
라면 맛에 대한 평가 자료를 예시로 만들어 PCA 를 이용해 분석했습니다.
면발, 그릇, 국물에 대해 각각 평가했습니다.



1. PC1은 국물과 면, 라면의 내용물에 대한 값이 되겠네요.
2. PC2는 외관에 대한 수치로 예상합니다.
3. 짬뽕라면이 종합적으로 맛이 가장 좋아보입니다.
4. 치즈라면이 생각보다 보통이네요.
5. 해물라면은 외관만 좋군요. 맛에 좀 더 신경써야겠습니다.
6. 얼큰라면은 맛에 비해 외관이 떨어지나봅니다.
7. 된장라면은 외관이 평균적인 데 비해 내용물이 형편없다네요.

3-ㄴ. 예시3 - 2018 러시아 월드컵 조별예션 경기결과

다음은 2018 러시아 월드컵 조별예선 경기결과를 biplot으로 나타낸 자료입니다.



승패가 색깔로 표시돼 있고 경기 내부 기록들이 있습니다.
Goals와 Offsides가 겹쳐있네요. Corner와 Shots 도요.

PC1은 양의 값이 공격적인 수치를 나타내고 있습니다. 슈팅, 유효슈팅, 코너, 점유율. 음의 값은 반대로 경고, 파울, 퇴장을 나타내고 있네요. 비교적 약팀일 것이고, 수비적으로 운영하면서 거친 플레이로 반칙을 많이 범했다고 보겠습니다.

PC2는 Goals 와 Offsides가 높은 수치를 나타내고 있는 것으로 보아 득점력으로 볼 수 있겠습니다. 수비 뒷공간을 파고들기 위해 많은 라인브레이킹 시도가 오프사이드로 이어졌고 골도 많이 난 것 같네요.

코너와 슈팅도 거의 같은 곳을 가리키고 있습니다. 그만큼 상관관계가 깊다고 보겠습니다. 실제 슈팅 시도마다 코너를 얻을 기회가 늘어나고, 코너마다 슈팅을 시도할 기회가 늘어나기 때문에 합리적으로 보입니다.

점유율이 골과 반대편에 위치하고 있는 것을 보면 공을 비효율적으로 점유했다고 볼 수 있습니다. 또는 상대적 약팀이 점유율을 내주고 고의적으로 수비를 강화하며 역습을 노렸다고 해석할 수도 있겠습니다.
승리팀이 가운데 몰려있는 것으로 보아 다방면에 충실하여 약점을 보완하도록 운영한 팀이 게임을 승리로 이끌었나봅니다.

더 얘기하면 너무 축덕처럼 보일까봐 줄이겠습니다. 이제 축구 안합니다.

축구는 아는 바가 조금이라도 있어서 더 많은 분석을 제공할 수 있었습니다. 역시 domain knowledge를 따라갈 순 없습니다.

파이썬 구현으로 넘어가기 전에 몇 가지만 더 살펴보고 가겠습니다.
많은 분들이 분석에 참고하실 거란 생각에 최대한 많은 것을 구겨담고 있습니다.

4. PCA의 또 다른 해석

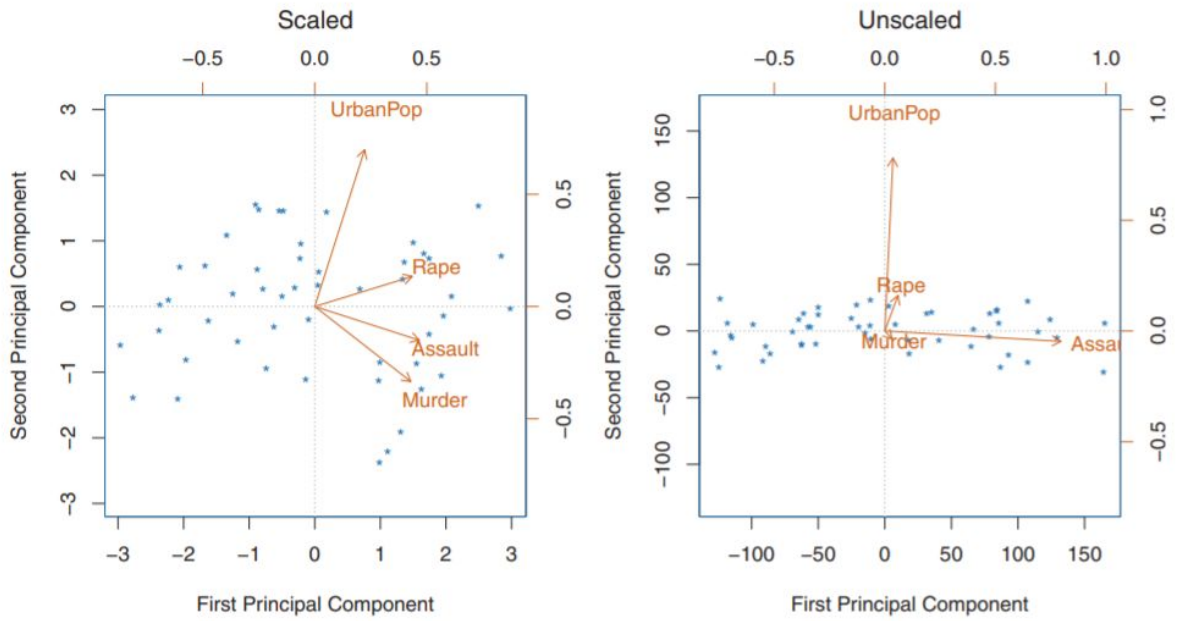
처음 말씀드렸던 게 기억나실지 모르겠습니다.
PCA loading vector 라인을 관측치로부터 가장 거리가 짧은 라인,
그러니까 가장 가까운 라인,
그러니까 데이터를 가장 잘 근사하는,
그러니까 정보를 가장 잘 간직한 라인
이라고 말씀드렸습니다.

2차원 데이터에서 1차원 선이었는데요.
그럼 3차원에서는 평면을 그리겠지요.
4차원 데이터에선 3차원 공간을 그릴 수 있겠습니다.
p차원에선 m차원의 가장 가까운 공간을 얻을 수 있을 것입니다.

정리하자면, PCA는 p차원의 데이터를 m차원의 가장 정보를 잘 보존하고 있는 근사데이터를 제공해줍니다.

5. 왜 scaling 해야하는가

scaling은 데이터를 일정 구간으로 동일하게 맞춰주는 것을 말합니다.
여기서는 평균을 0, 분산이 1이 되도록 scaling 해준다고 했죠.
안 하면 어떻게 되는지 잠시 보겠습니다.



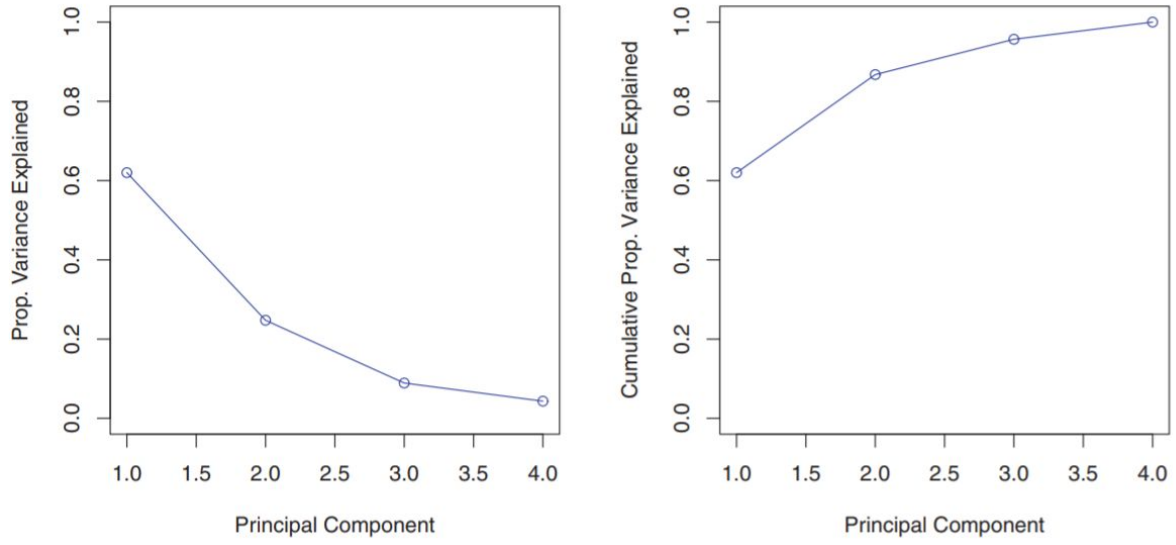
왼쪽 그림은 scaling 해서 PCA했던, 이미 우리가 본 결과입니다.
오른쪽 그림은 unscaled 자료에서 PCA 시행한 결과입니다.

Assault 의 loading이 매우 크네요.
이는 각 feature별 분산이 다르기 때문에 계산결과가 달라져서인데요.
분산이 각각 약 18, 87, 6945, 209 입니다. Assault의 분산이 큰만큼 특이한 결과가 나오네요.
구체적인 식은 생략하고, 항상 scale 이후에 계산하도록 합니다.

6. Proportion of Variance Explained(PVE)

지금까지 우리는 첫 번째와 두 번째 주성분을 봐왔습니다.
그렇다면 아주 자연스럽게 궁금하실 것이,

이 두 주성분으로 project 하면서 얼마나 많은 정보를 잃었을까
그러니까 첫 두 주성분에 포함되지 않은 분산이 얼마일까,
그러니까 주성분으로 얼마나 많은 분산이 설명되는가가 궁금합니다.



위 그래프는 USArrests에서 PVE를 나타낸 scree plot 입니다.

왼쪽은 PC1, PC2, PC3, PC4로 넘어갈 때마다 각각 분산이 얼마나 설명됐는지입니다.
오른쪽은 Cumulative 보이시나요? 누적 PVE입니다.
PC1으로 데이터의 62% 가량의 분산이 설명되었다네요.
PC2로는 24.7%, PC3는 13% 입니다.
첫 두 주성분만으로 데이터의 86%를 요약해 보여줬던 거였군요.

그렇다면 또 자연스러운 질문이 이어집니다.
이전에 min(n-1, p) 만큼 주성분을 구한다고 했는데요. 보통 그 모든 주성분에 관심이 있진 않습니다.
최소한의 주성분만으로 데이터를 시각화하고 분석하면 좋지 않을까요?
첫 두 주성분만 가지고 하면 얼마나 편할까요.
데이터를 잘 표현하려면 몇 개의 주성분이 필요한 걸까요?

정해진 답은 없지만

1. PVE 수치를 보고 적당한(!) 선에서 끊습니다. USArrests 같은 경우 첫 두 주성분만으로 86%가 나오니 PC2에서 끊어도 되겠습니다. scree plot 에서 꺾이는 부분을 보고 정한다고 해서 elbow 라고도 합니다.

2. 흥미로운 패턴이 발견될 때까지 찾습니다. 첫 두 세개 주성분으로 의미있는 패턴이 발견된다면 거기서 끝내지만, 아니라면 주성분을 더 구해 패턴을 찾아보기도 합니다.

양이 많아졌습니다. 이론은 여기까지 하고 파이썬 코드와 태블로 대시보드는 PCA 실습에서 다루겠습니다.

감사합니다.