Wróżenie z punktów

ordynacja w eksploracji danych

Marcin K. Dyderski

27 września 2017

Spis treści

1. Czym jest ordynacja

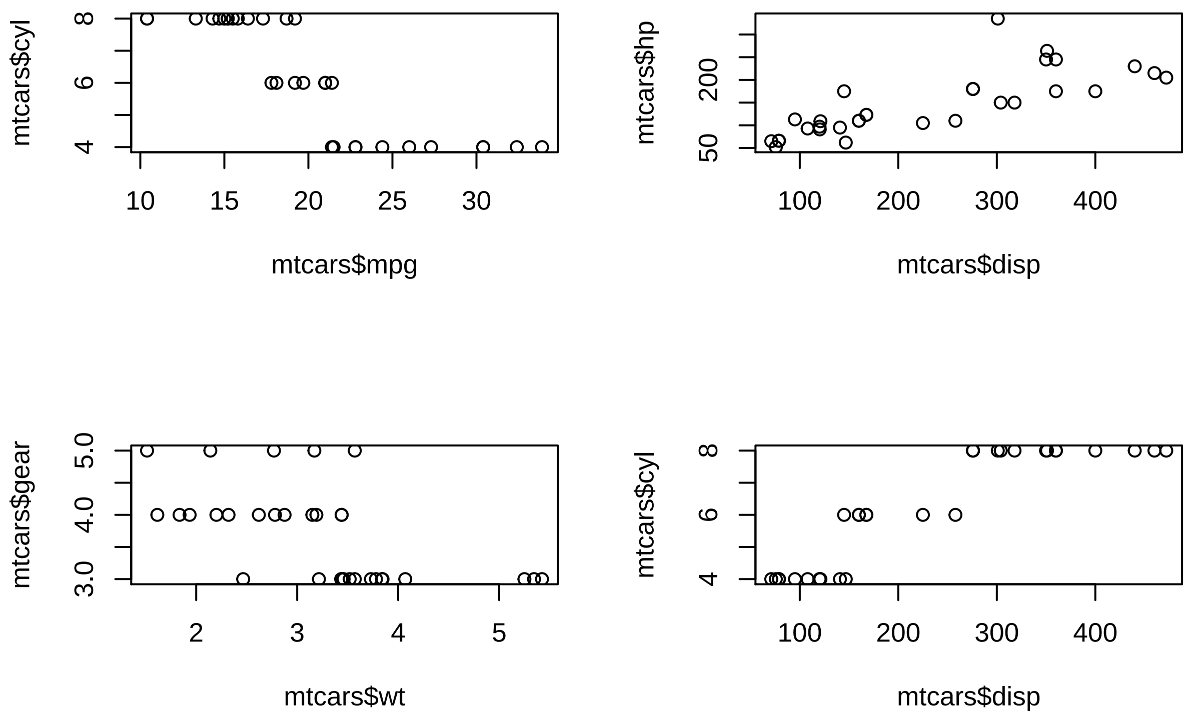
Ordynacja to metoda analizy danych polegająca na uporządkowaniu danych o wielu cechach. Porządkowanie to odbywa się wzdłuż tylu osi, ile cech jest analizowane. Zobaczmy sobie na standardowy zbiór danycgh `mtcars`

```
summary(mtcars)
```

```
##       mpg             cyl             disp             hp       
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0  
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
##       drat             wt             qsec             vs        
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
##        am              gear            carb      
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000  
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
##  Median :0.0000   Median :4.000   Median :2.000  
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812  
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000  
```

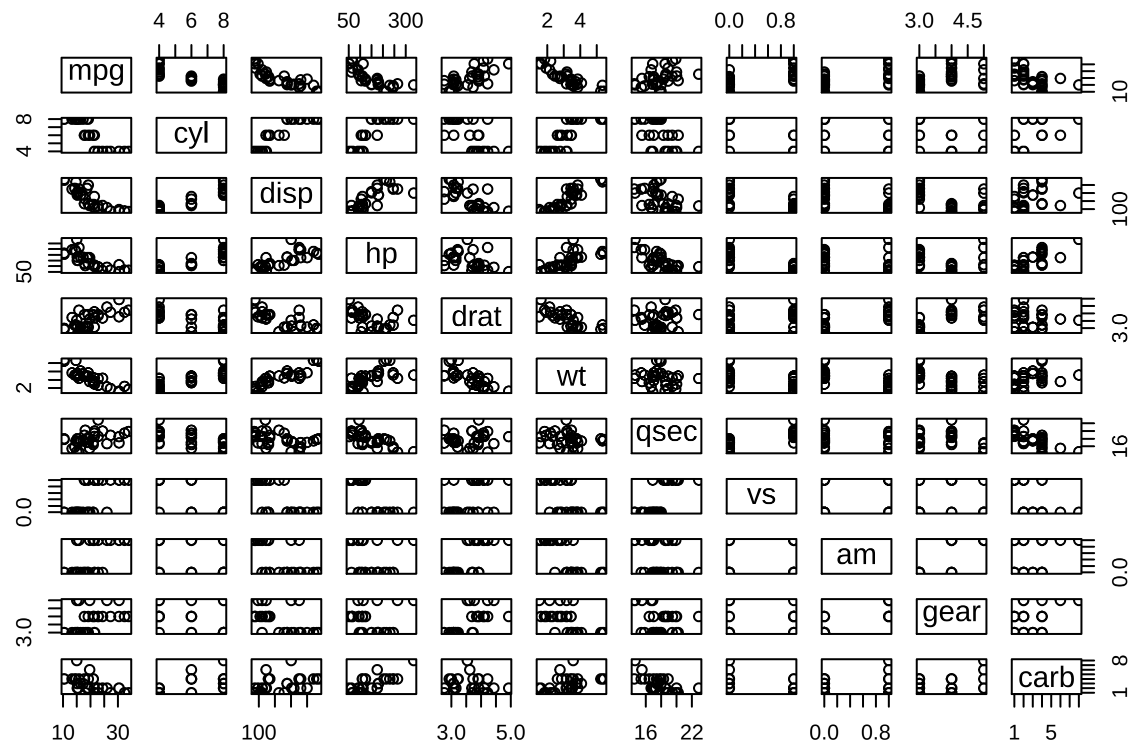
W tym zbiorze danych mamy 11 zmiennych. Możemy zobaczyć jak punkty rozkładają się wdłuż poszczególnych cech:

```r
par(mfrow=c(2,2))
plot(mtcars$mpg,mtcars$cyl)
plot(mtcars$disp,mtcars$hp)
plot(mtcars$wt,mtcars$gear)
plot(mtcars$disp,mtcars$cyl)
```



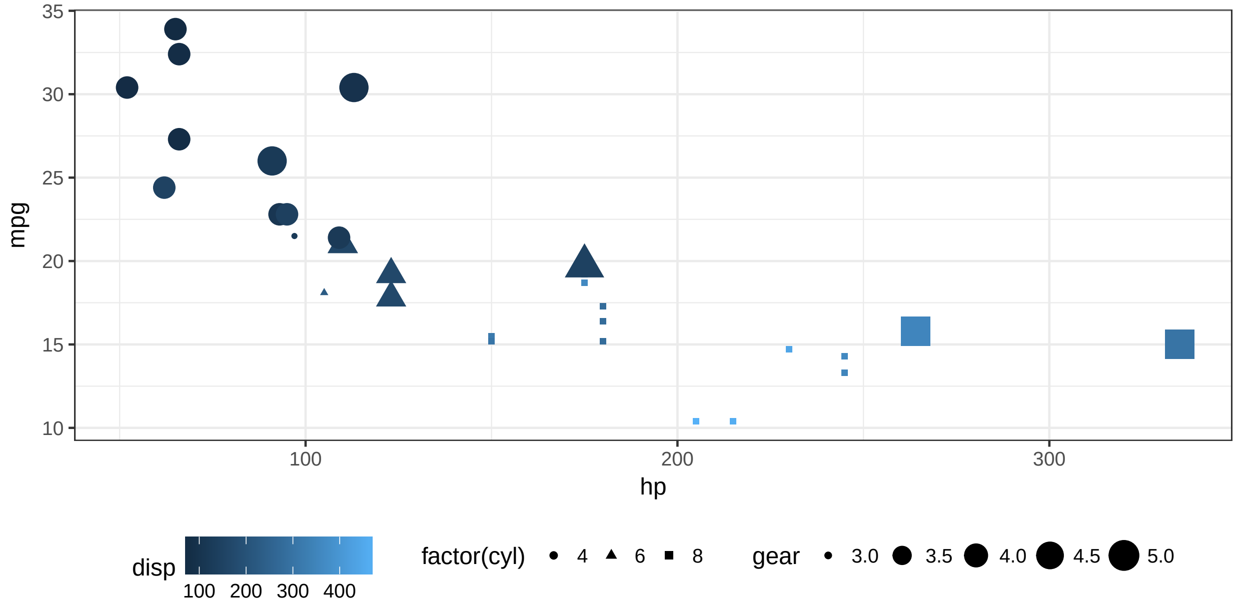
Możemy spróbować także obejrzeć te dane za pomocą funkcji `pairs()`

```
par(mfrow=c(1,1))
pairs(mtcars)
```



Na wykresie można spróbować umieścić więcej niż dwie zmienne, np.:

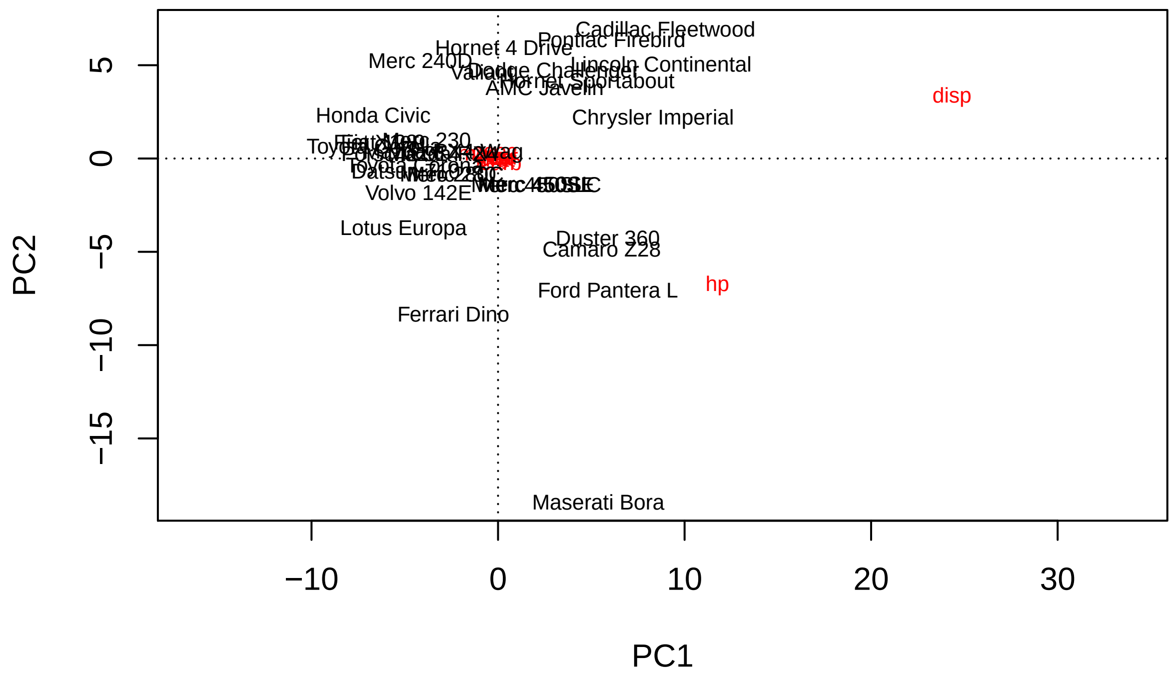
```
library(ggplot2)
g<-ggplot(mtcars, aes(x=hp,y=mpg,col=disp, shape=factor(cyl),size=gear))+geom_point()
g+theme_bw()+theme(legend.position = 'bottom')
```



Jednak nie widzimy wszystkich relacji pomiędzy zmiennymi, sam obraz zaś jest mało czytelny i nieintuicyjny.

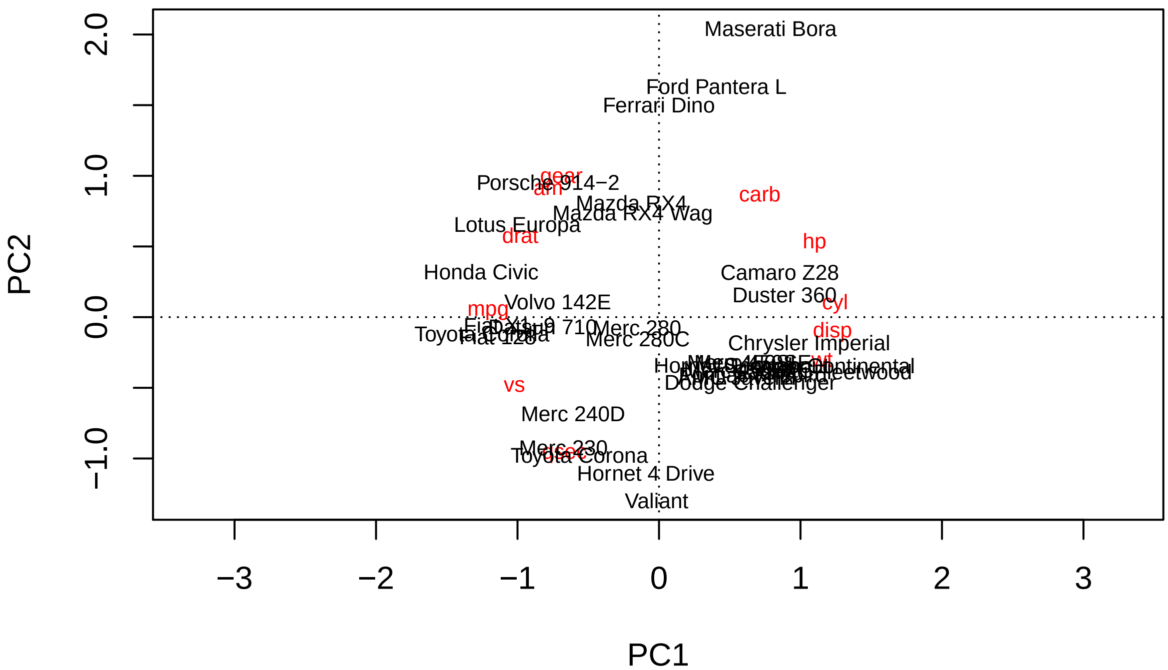
Z pomocą może nam przyjść jedna z metod ordynacji - analiza głównych składowych (ang. Principal Components Analysis, *PCA*):

```
library(vegan)
pca1<-rda(mtcars)
plot(pca1)
```



Na pierwszym, roboczym obrazku widzimy już że podobne samochody grupują się obok siebie, a cechy (zaznaczone kolorem czerwonym) są w większości przypadków zgrupowane obok siebie. Patrząc na zakres wartości liczbowych w `summary(mtcars)` widzimy że dwie wyraźnie wyznaczone cechy mają szersze zakresy wartości. Wystandaryzujmy więc dane przed analizą:

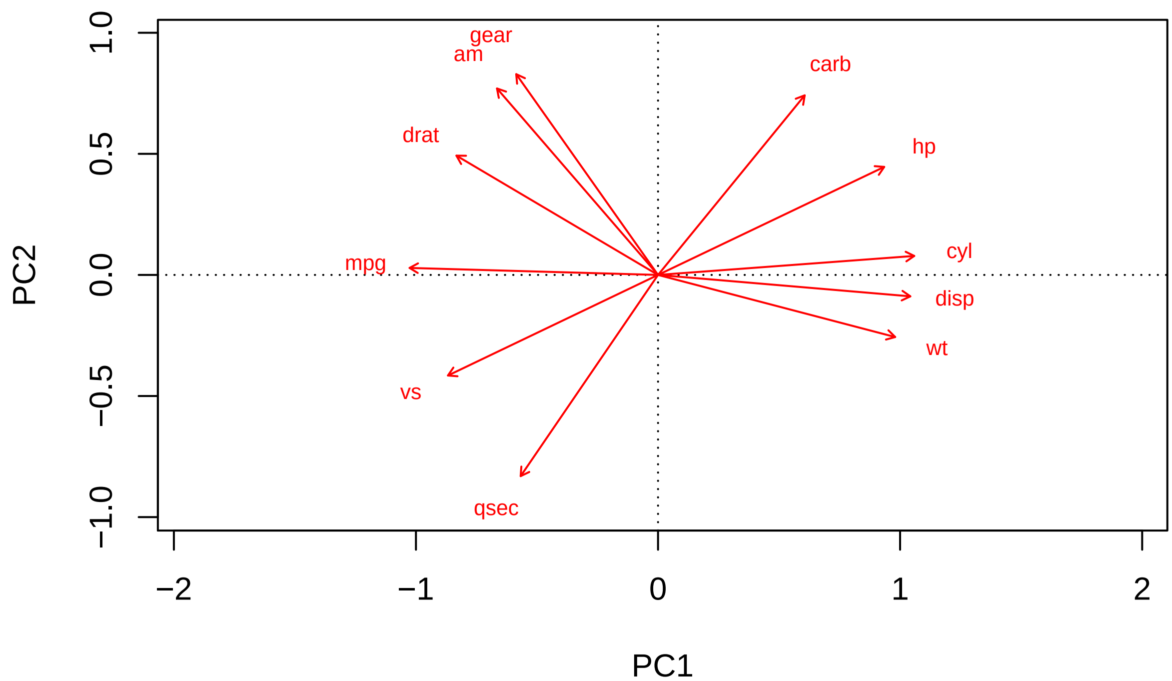
```
pca2<-rda(decostand(mtcars, method='standardize'))
plot(pca2)
```



Widzimy już bardziej jakieś zależności. W dalszej części będziemy mówić o cechach i obiektach - w pakiecie `vegan` są one nazywane odpowiednio 'species' i 'sites'. Wynika to z faktu, że pakiet ten został stworzony do analiz roślinności. Argumentem funkcji wykonujących ordynacje jest data.frame, w którym wiersze to obiekty, a cechy - kolumny.

Zostawmy tylko cechy samochodów:

```
biplot(pca2, display='species')
```



Po prawej stronie przestrzeni analitycznej mamy trzy zmienne których wartości rosną wraz ze wzrostem składowej PC1 - `cyl`, `disp` oraz `wt`, czyli możemy na tej podstawie wnioskować, że wraz ze wzrostem pojemności skokowej `disp` rośnie masa `wt` oraz liczba cylindrów silnika `cyl`. Strzałka wskazująca na moc `hp` jest skierowana bliżej osi PC1 niż PC2, więc zależność mocy od tych cech będzie słabsza. Strzałka `mpg` jest skierowana odwrotnie do `cyl` oraz `disp`, co wskazuje na to, że im więcej cylindrów i większa pojemność skokowa, tym mniej mil przejedziemy zużywając jeden galon paliwa `mpg`. Dzięki takiemu przedstawieniu zmiennych mamy lepszy ogląd danych i zależności pomiedzy nimi. Takie wyniki wydają nam się intuicyjne. Pytanie, czy tego typu analizy można wykorzystać do czegoś innego niż wstępny ogląd danych przed właściwą pracą analityczną?

Przykłady zastosowania PCA można znaleźć np. w Nature (link), gdzie wykorzystano sześć cech 46 tysięcy gatunków roślin z całej kuli ziemskiej.

2. Podział metod, zastosowania i przykłady

Głównymi czynnikami różnicującymi metody ordynacji jest możliwość dodania innych zmiennych jako aktywne wektory wpływające na rozrzut punktów w przestrzeni ordynacyjnej oraz rodzaj gradientów w danych. Najczęściej używamy podziału na analizy dostosowane do krótkich gradientów (liniowych) oraz długich (unimodalnych). Jako gradienty rozumiemy zmienność poszczególnych cech wzdłuż całego zbioru danych. Drugim podziałem jest możliwość włączenia dodatkowych cech jako aktywne wskaźniki ograniczające rozrzut punktów w przestrzeni. Najczęściej są to wskaźniki o innej wadze lub znaczeniu, np. jeśli analizujemy występowanie gatunków zwierząt to takim czynnikiem może być np. lesistość lub parametry klimatyczne. Mówimy wtedy o analizach ograniczonych lub pośrednich (ang. *constrained*).

metoda	krótkie gradienty	długie gradienty
nieograniczone	PCA i CA	CA i DCA
ograniczone	RDA	CCA

podział za Zelenym link

CA wg Zelenego jest do dłuższych gradientów niż PCA, jednak częściej bywa zestawiana razez z PCA. Przy dłuższych gradientach zarówno CA jak i PCA tworzą artefakty i konieczne jest użycie DCA.

PCA - Principal Components Analysis

PCA to metoda o największej elegancji pod względem obliczeniowym. Opiera się o wyliczenie głównych składowych, czyli sztucznych zmiennych, będących liniowymi kombinacjami analizowanych cech, jak najściślej skorelowanych z nimi skorelowanymi. Najczęściej kilka głównych składowych pozwala wyjaśnić większość zmienności zbioru danych. W pakiecie `vegan` do wykonania PCA wykorzystujemy funkcję `rda()`. Popatrzmy na analizowany przed chwilą przykład:

```
pca1<-rda(mtcars)
pca1
```

```
## Call: rda(X = mtcars)
##
##               Inertia Rank
## Total           20109
## Unconstrained   20109   11
## Inertia is variance
##
## Eigenvalues for unconstrained axes:
##   PC1   PC2   PC3   PC4   PC5   PC6   PC7   PC8   PC9  PC10  PC11
## 18641  1455     9     2     1     0     0     0     0     0     0
```

wywołanie obiektu pca1 pokazuje nam inercję analizy (Inertia) oraz wartości własne dla poszczególnych osi - składowych PCA. Całkowita inercja układu wynosi 20109, natomiast wartość własna osi PC1 - 18641. Oznacza to, że oś PC1 wyjaśnia 18641/20109 czyli 92,7% zmienności układu, oś PC2 natomiast 1455/20109, czyli 7,2%. Widzieliśmy natomiast że mimo tak dobrych parametrów wynik jest nieczytelny, co wynika z dużych wartości dwóch parametrów - `hp` oraz `dist`. Z tego powodu wykonaliśmy standardyzowanie za pomocą funkcji `decostand()`. Potrzebne będą nam dwa argumenty - obiekt który będziemy analizować (`data.frame`) oraz `method` - jedna z kilku dostępnych wg opisu w pomocy `?decostand`. Po wpisaniu złej nazwy metody funkcja zwraca nam podpowiedzi;) Spójrzmy na parametry tej drugiej analizy:

```r
pca2<-rda(decostand(mtcars, method='standardize'))
pca2
```

```
## Call: rda(X = decostand(mtcars, method = "standardize"))
##
##                Inertia Rank
## Total               11
## Unconstrained       11   11
## Inertia is variance
##
## Eigenvalues for unconstrained axes:
##   PC1   PC2   PC3   PC4   PC5   PC6   PC7   PC8   PC9  PC10  PC11
## 6.608 2.650 0.627 0.270 0.223 0.212 0.135 0.123 0.077 0.052 0.022
```

Tutaj całkowita inercja układu wynosi 11, pierwsza oś wyjaśnia 60% zmienności, druga - 24%. Jest to wynik bardzo zadowalający - przy dużych zbiorach danych o dużej zmienności nawet wyjaśnienie 20% zmienności może być satysfakcjonujące. Wartości poszczególnych osi dla cech oraz obiektów można wyciągnąć za pomocą przeciążonej funkcji `summary()` która dla obiektu typu `rda` lub `cca` zwraca zestawienie

```r
summary(pca2)
```

```
##
## Call:
## rda(X = decostand(mtcars, method = "standardize"))
##
## Partitioning of variance:
##                Inertia Proportion
## Total               11          1
## Unconstrained       11          1
##
## Eigenvalues, and their contribution to the variance
##
## Importance of components:
##                          PC1    PC2     PC3     PC4     PC5     PC6    PC7
## Eigenvalue            6.6084 2.6505 0.62720 0.26960 0.22345 0.21160 0.1353
## Proportion Explained  0.6008 0.2409 0.05702 0.02451 0.02031 0.01924 0.0123
## Cumulative Proportion 0.6008 0.8417 0.89873 0.92324 0.94356 0.96279 0.9751
##                           PC8     PC9    PC10    PC11
## Eigenvalue            0.12290 0.07705 0.05204 0.02204
## Proportion Explained  0.01117 0.00700 0.00473 0.00200
## Cumulative Proportion 0.98626 0.99327 0.99800 1.00000
##
## Scaling 2 for species and site scores
## * Species are scaled proportional to eigenvalues
## * Sites are unscaled: weighted dispersion equal on all dimensions
## * General scaling constant of scores:  4.29723
##
##
## Species scores
##
##          PC1      PC2      PC3       PC4      PC5      PC6
## mpg  -1.2075  0.03401 -0.23164 -0.015164  0.06299 -0.06484
## cyl   1.2454  0.09227 -0.17989 -0.001744  0.03582  0.10046
## disp  1.2263 -0.10404 -0.06309  0.172632  0.24131 -0.20035
## hp    1.0993  0.52478  0.14367 -0.045529  0.33076  0.04258
```

```
## drat -0.9797  0.57943  0.16540  0.575081  0.04736  0.14572
## wt    1.1528 -0.30172  0.35074  0.165427 -0.04595 -0.27710
## qsec -0.6677 -0.97743  0.41370  0.045798 -0.10085 -0.19697
## vs   -1.0209 -0.48863  0.44001 -0.144538  0.36720  0.11563
## am   -0.7825  0.90580 -0.21114 -0.020494  0.05499 -0.34021
## gear -0.6892  0.97527  0.29735 -0.178069  0.02960 -0.14516
## carb  0.7128  0.87238  0.54235 -0.085297 -0.22130  0.10938
##
##
## Site scores (weighted sums of species scores)
##
##                            PC1      PC2     PC3       PC4       PC5
## Mazda RX4           -0.1942101  0.80977 -0.5767  0.169012 -1.543792
## Mazda RX4 Wag       -0.1859898  0.72326 -0.3667  0.295983 -1.659973
## Datsun 710          -0.8213269 -0.06834 -0.2314 -0.364500  0.651074
## Hornet 4 Drive      -0.0921299 -1.10260 -0.1302 -0.748873  0.896714
## Hornet Sportabout    0.5834722 -0.35201 -1.0881  0.110684  0.338819
## Valiant             -0.0165889 -1.29997  0.1571 -1.449536  0.345594
## Duster 360           0.8873065  0.15626 -0.3480 -0.076596  0.561413
## Merc 240D           -0.6073608 -0.68367  0.9054 -0.211268 -0.517009
## Merc 230            -0.6759416 -0.92553  1.7239  0.426924 -0.544816
## Merc 280            -0.1555485 -0.07560  1.4319  0.098497 -0.113678
## Merc 280C           -0.1504730 -0.15113  1.6149  0.140257 -0.242957
## Merc 450SE           0.6642401 -0.31891 -0.3601 -0.192938 -0.618173
## Merc 450SL           0.6051427 -0.31880 -0.4647 -0.313627 -0.580621
## Merc 450SLC          0.6349058 -0.37410 -0.2831 -0.260623 -0.705572
## Cadillac Fleetwood   1.1524092 -0.38633  0.6209  0.431822 -0.078772
## Lincoln Continental  1.1684648 -0.34220  0.6912  0.602512  0.006366
## Chrysler Imperial    1.0617427 -0.19650  0.5265  0.989477  0.339654
## Fiat 128            -1.1395527 -0.13847 -0.4056  0.082039  0.359172
## Honda Civic         -1.2570896  0.32122 -0.1984  1.735467  0.159476
## Toyota Corolla      -1.2512351 -0.13032 -0.4472  0.272485  0.362716
## Toyota Corona       -0.5626919 -0.98913  0.1504  0.075087  0.064165
## Dodge Challenger     0.6456352 -0.47324 -1.1211 -0.869546 -0.369387
## AMC Javelin          0.5506399 -0.42296 -0.9232  0.008464 -0.412373
## Camaro Z28           0.8537135  0.31768 -0.1565  1.210475  0.635329
## Pontiac Firebird     0.6636812 -0.40773 -1.0018  0.217647  0.488616
## Fiat X1-9           -1.0561270 -0.05655 -0.4351 -0.019959  0.337574
## Porsche 914-2       -0.7834602  0.95485 -0.7965  0.845142 -0.975257
## Lotus Europa        -1.0004944  0.64327 -0.4353 -1.714169  1.134210
## Ford Pantera L       0.4057163  1.63313 -0.1310  0.877151  1.798703
## Ferrari Dino        -0.0002925  1.50205  0.3857 -1.395676 -1.385923
## Maserati Bora        0.7887411  2.04359  1.2977 -1.304110  0.743329
## Volvo 142E          -0.7152984  0.10902  0.3950  0.332294  0.525378
##                           PC6
## Mazda RX4           -0.028502
## Mazda RX4 Wag       -0.405578
## Datsun 710          -0.585181
## Hornet 4 Drive       0.032378
## Hornet Sportabout    0.250324
## Valiant             -0.409121
## Duster 360           1.195793
## Merc 240D           -0.001659
## Merc 230            -0.560185
```

```
## Merc 280             1.370019
## Merc 280C            1.226239
## Merc 450SE           0.220976
## Merc 450SL           0.402729
## Merc 450SLC          0.302349
## Cadillac Fleetwood  -1.484017
## Lincoln Continental -1.447294
## Chrysler Imperial   -1.096720
## Fiat 128            -0.784530
## Honda Civic          0.869221
## Toyota Corolla      -0.532135
## Toyota Corona        1.214261
## Dodge Challenger     0.178219
## AMC Javelin          0.484581
## Camaro Z28           1.588725
## Pontiac Firebird    -0.332770
## Fiat X1-9           -0.243273
## Porsche 914-2       -0.569508
## Lotus Europa         0.027691
## Ford Pantera L      -0.292979
## Ferrari Dino        -0.016371
## Maserati Bora       -0.026190
## Volvo 142E          -0.547488
```

Ponieważ pakiet `vegan` powstał do pracy z danymi o roślinności to w dalszej części mówiąc o analizowanych cechach zbioru danych odnosić będziemy się do 'species scores', a mówiąc o obiektach - do 'site scores'. Współrzędne dla poszczególnych osi można wyekstrachować także za pomocą funkcji `scores()`, np.:

```r
scores(pca2)
```

```
## $species
##             PC1         PC2
## mpg  -1.2074940  0.03401238
## cyl   1.2454162  0.09227184
## disp  1.2263283 -0.10404301
## hp    1.0993330  0.52477847
## drat -0.9797411  0.57943248
## wt    1.1527793 -0.30172112
## qsec -0.6676675 -0.97743065
## vs   -1.0209088 -0.48863007
## am   -0.7825331  0.90580230
## gear -0.6891837  0.97526605
## carb  0.7128367  0.87237591
##
## $sites
##                              PC1         PC2
## Mazda RX4           -0.1942100570  0.80977314
## Mazda RX4 Wag       -0.1859897769  0.72325826
## Datsun 710          -0.8213268623 -0.06833785
## Hornet 4 Drive      -0.0921299310 -1.10260397
## Hornet Sportabout    0.5834721641 -0.35201022
## Valiant             -0.0165889449 -1.29997016
## Duster 360           0.8873064962  0.15626120
## Merc 240D           -0.6073607585 -0.68366532
## Merc 230            -0.6759415521 -0.92552965
```

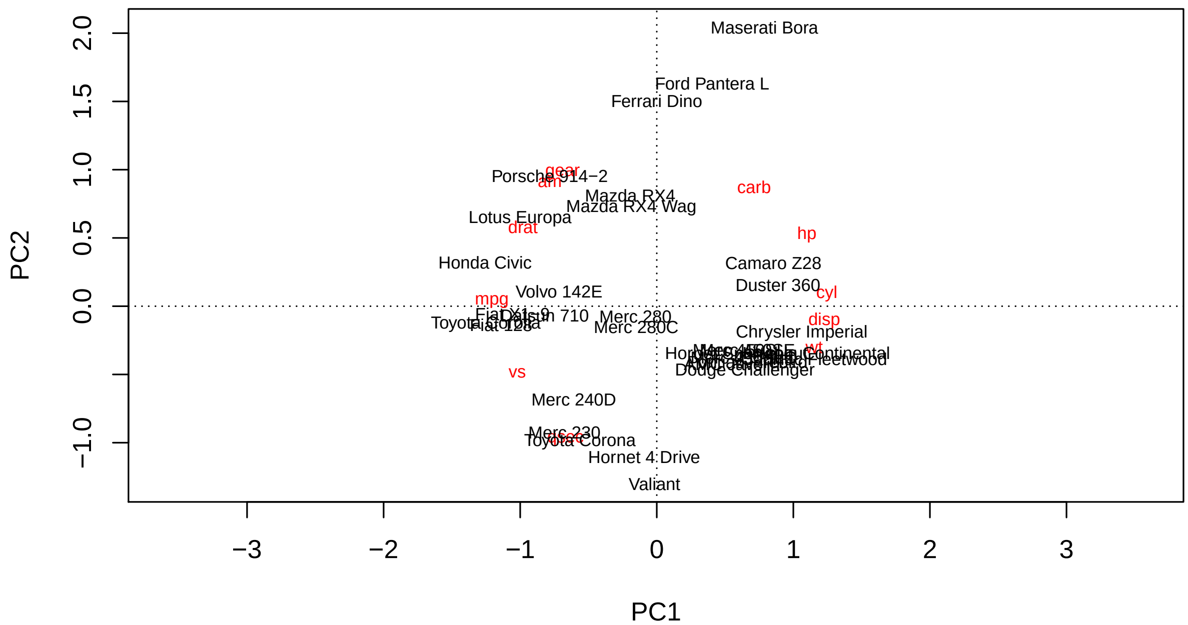
```
## Merc 280            -0.1555484945 -0.07559639
## Merc 280C           -0.1504729780 -0.15113180
## Merc 450SE           0.6642401442 -0.31891454
## Merc 450SL           0.6051427047 -0.31879635
## Merc 450SLC          0.6349058321 -0.37409819
## Cadillac Fleetwood   1.1524091523 -0.38632733
## Lincoln Continental  1.1684647703 -0.34220178
## Chrysler Imperial    1.0617426601 -0.19650497
## Fiat 128            -1.1395527121 -0.13846685
## Honda Civic         -1.2570896185  0.32121959
## Toyota Corolla      -1.2512351364 -0.13031781
## Toyota Corona       -0.5626919029 -0.98913386
## Dodge Challenger     0.6456352143 -0.47324200
## AMC Javelin          0.5506398857 -0.42296375
## Camaro Z28           0.8537135255  0.31767899
## Pontiac Firebird     0.6636811932 -0.40772787
## Fiat X1-9           -1.0561269663 -0.05655469
## Porsche 914-2       -0.7834602115  0.95485335
## Lotus Europa        -1.0004944358  0.64326567
## Ford Pantera L       0.4057163463  1.63312838
## Ferrari Dino        -0.0002925269  1.50204536
## Maserati Bora        0.7887411330  2.04359312
## Volvo 142E          -0.7152983562  0.10901829
##
## attr(,"const")
## [1] 4.29723
```

```r
scores(pca2, choices = c(3,4))$species
```

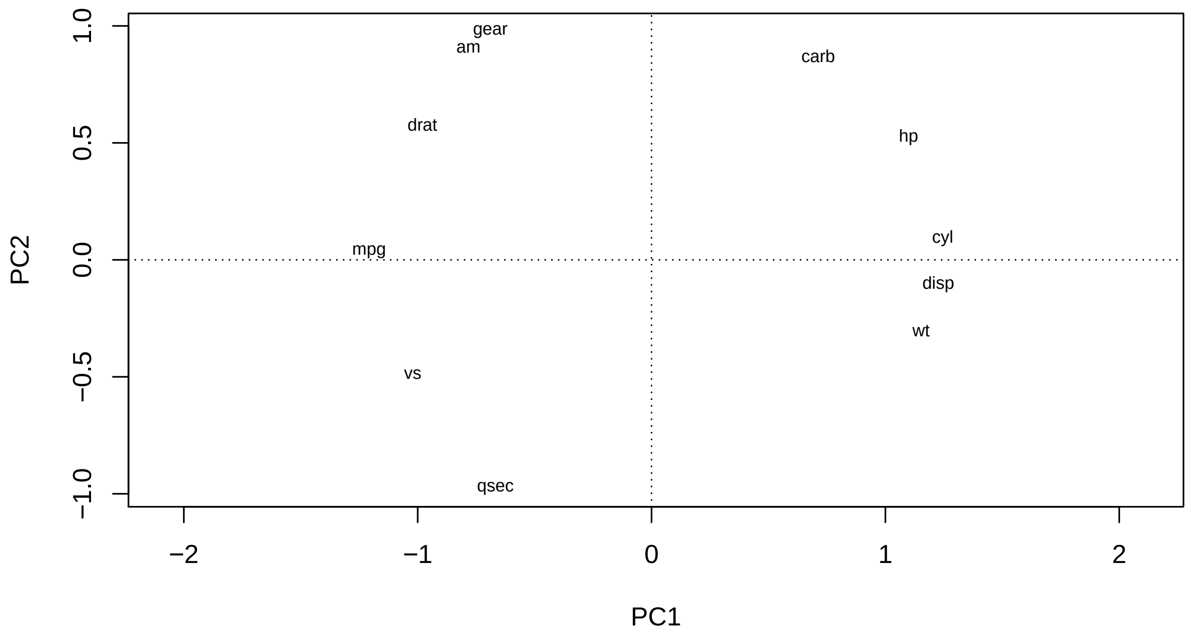
```
##              PC3          PC4
## mpg  -0.23163859 -0.015163831
## cyl  -0.17988872 -0.001743644
## disp -0.06308955  0.172631533
## hp    0.14367068 -0.045528760
## drat  0.16539759  0.575081302
## wt    0.35074372  0.165427402
## qsec  0.41369617  0.045798110
## vs    0.44001195 -0.144538217
## am   -0.21113933 -0.020493753
## gear  0.29734637 -0.178069082
## carb  0.54234540 -0.085296718
```

argument `choices` wskazuje które osie analityczne należy wybrać, domyślnie jest to oś 1 i 2. Za pomocą operatora `$` można wybrać cechy lub obiekty (`species` lub `sites`). Do wywołania wykresu pokazującego wynik analizy możemy użyć funkcji `plot()`, w argumencie `display` podając czy chcemy wyświetlić współrzędne cech `display='species'` czy obiektów `display='sites'`:

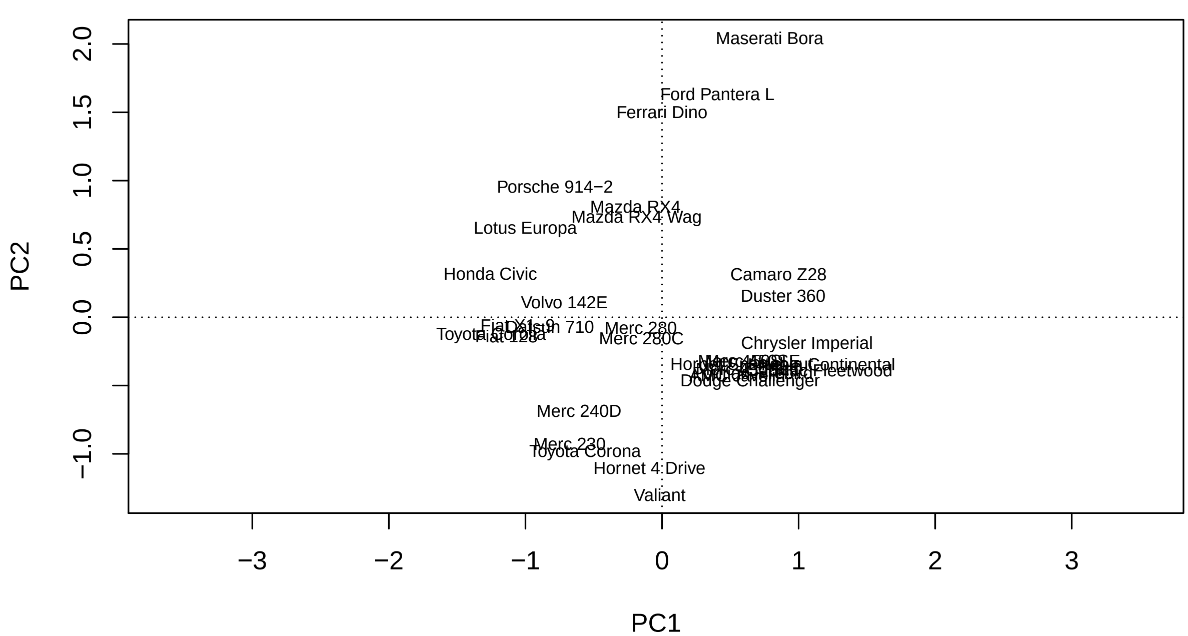
```
plot(pca2)
```




```
plot(pca2, display='species')
```



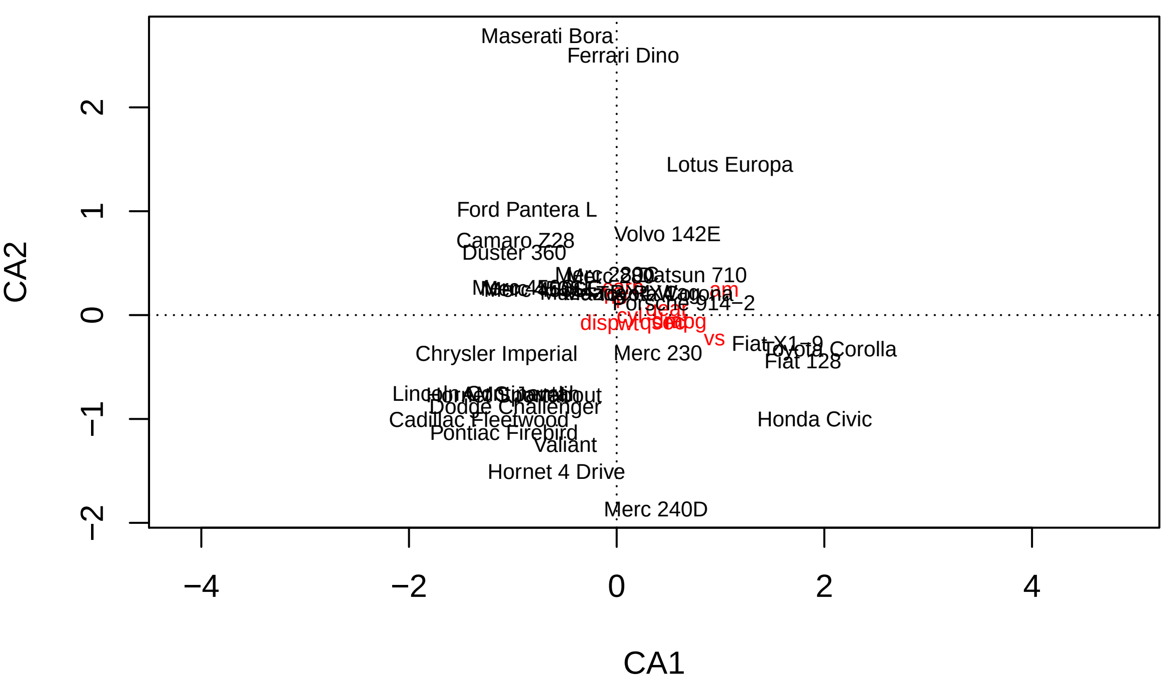

```r
plot(pca2, display='sites')
```



CA Correspondence Analysis

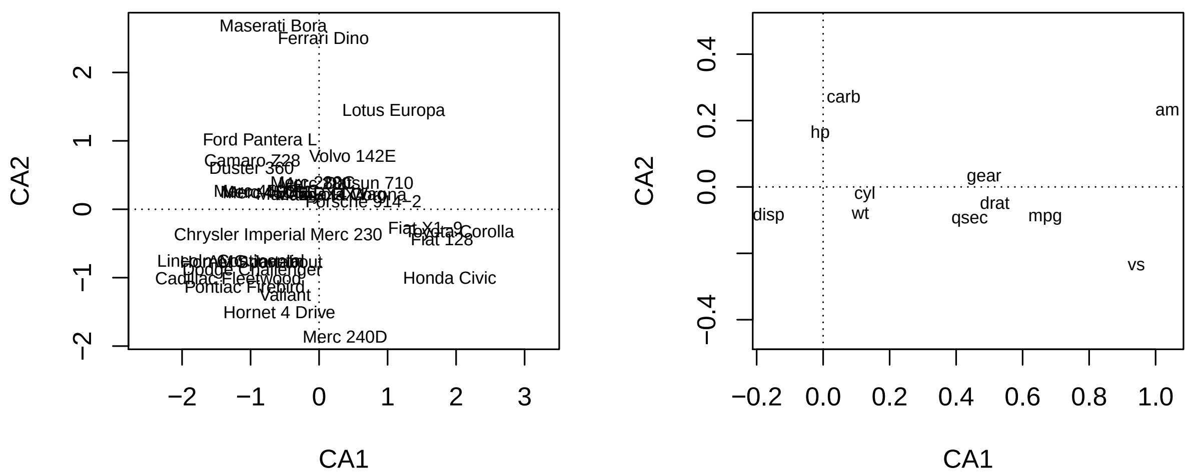
CA to metoda przeznaczona do krótkich gradientów z unimodalnym rozkładem cech. Różni się ona od PCA algorytmem obliczania punktów - tutaj nie wyznacza się głównych składowych, lecz dla obiektów przyjmuje się arbitralnie wskaźniki (wartości). Następnie dla cech oblicza się średnie ważone tych wartości, gdzie wagą są wartości cech dla poszczególnych obiektów. Kolejnym krokiem jest ustalenie nowych wartości dla obiektów i iteracyjne ustalanie takich wartości, by wskaźniki dla prób się nie zmieniały. Takie wzajemne uśrednianie dało nazwę - analiza zgodności lub analiza korespondencji. Do wykonania tej analizy wykorzystuje się funkcję `cca()`:

```
#ca1<-cca(decostand(mtcars, method='standardize'))
#nie działa
ca1<-cca(decostand(mtcars, method='normalize'))
plot(ca1)
```



analiza CA nie obsługuje wartości ujemnych, stąd nie można danych wystandardyzować, lecz stosujemy inną transformację - skalowanie (*vide* `?decostand`). Widzimy inny niż w przypadku PCA układ punktów, jednak odstające modele są dalej podobne. Zobaczmy to wyraźniej

```r
par(mfrow =c(1,2))
plot(ca1, display='sites')
plot(ca1, display='species')
```



Dla obiektów i dla cech układ wygląda nieco inaczej niż przy PCA. szczególnie widać tu różne znaczenie cech - im dalej od centrum układu współrzędnych, tym większy wpływ danej cechy.

DCA Detrended Correspondence Analysis

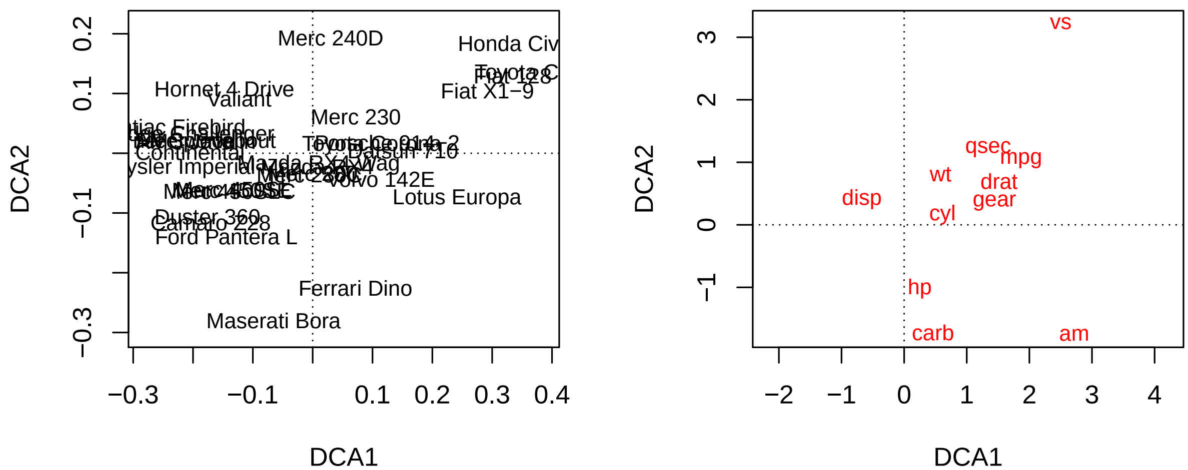
DCA została opracowana do analizy danych z szerokich zakresów zmienności cech - długich gradientów. Algorytm CA został wzbogacony o kolejny krok - dopasowanie wskaźników korygujących efekt łuku lub efekt podkowy - problem występujący w przypadku CA i PCA, gdy gradient jest zbyt długi i punkty są "upakowywane" w danej osi, przez co sprawiają wrażenie mniejszych różnic niż w rzeczywistości. Zastosowanie wskaźników korygujących powoduje że w przeciwieństwie do pozostałych metod współrzędne DCA wyrażone są w jednostkach odchyleń standardowych zamiast w jednostkach abstrakcyjnych. DCA jest wykorzystywane do wstępnej oceny długości gradientów. Przyjmuje się że jeśli długość pierwszej osi jest większa niż 3 to powinno się stosować DCA, natomiast jeśli mniejsza niż 2 to można stosować PCA i CA. Inne wskazówki mówią o wartościach 3 i 4. Aby wykonać DCA należy skorzystać z funkcji `decorana()`:

```r
dca1<-decorana(decostand(mtcars, method='normalize'))
dca1
```

```
##
## Call:
## decorana(veg = decostand(mtcars, method = "normalize"))
##
## Detrended correspondence analysis with 26 segments.
## Rescaling of axes with 4 iterations.
##
##                     DCA1    DCA2     DCA3     DCA4
## Eigenvalues      0.05814 0.01732 0.016341 0.016378
## Decorana values  0.06150 0.00970 0.001974 0.001277
## Axis lengths     0.66659 0.47304 0.463953 0.462383
```

W tym przypadku otrzymujemy inne podsumowanie analizy - mamy długości osi, które mówią nam o tym, że tutaj gradienty są krótkie i uprawniają do zastosowania CA lub CCA. Eigenvalues to wartości własne - tutaj są już wyskalowane i wskazują jaką część zmienności wyjaśnia dana oś - ta analiza wyjaśnia bardzo mało zmienności. Decorana values to wartości własne wyliczone wg innego algorytmu - autorzy pakietu nie zalecają korzystania z nich. Do wyświetlania wyników używamy tak samo funkcji `plot()`:

```
par(mfrow =c(1,2))
plot(dca1, display='sites')
plot(dca1, display='species')
```



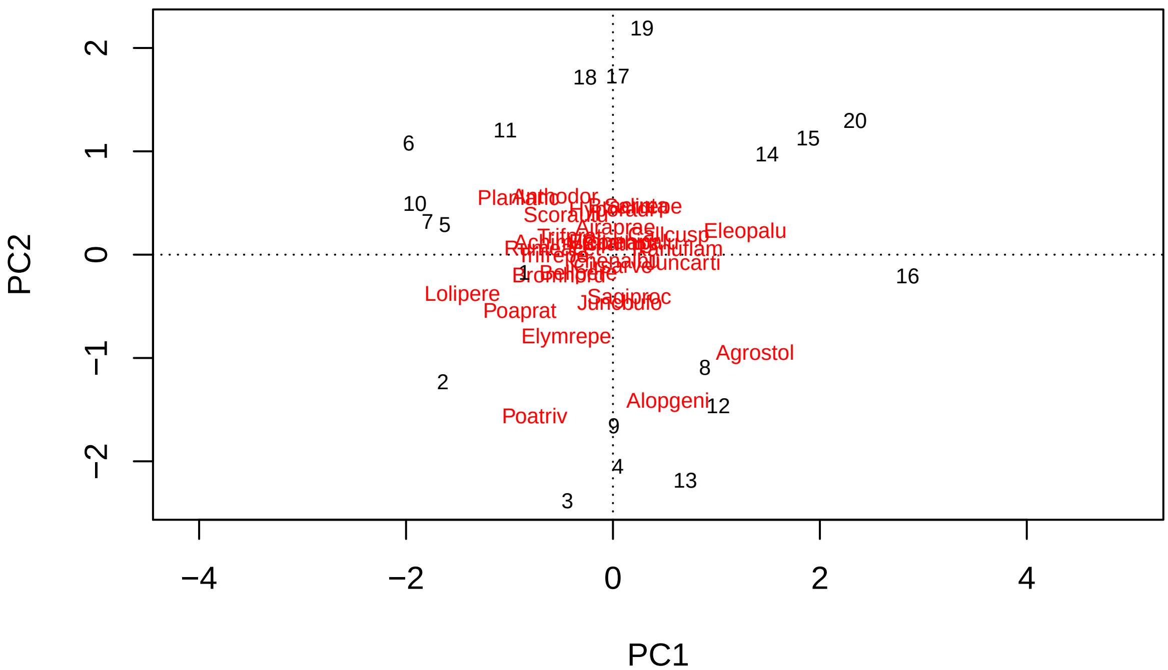
RDA - Redundancy Analysis

RDA jest ograniczoną wersją PCA. Do zestawu danych możemy dołączyć inny - np. opisujący czynniki mogące wpływać na analizowane cechy. Przyjrzyjmy się zestawowi danych z pakietu `vegan` o nazie `dune` - opisującym roślinność wydm. Tutaj cechami są gatunki roślin, których nazwy nie mają żadnego znaczenia, natomiast obiektami - poletka badawcze. Możemy zrobić na tym zwykłe PCA:

```
data(dune)
pca.dune<-rda(dune)
pca.dune
```

```
## Call: rda(X = dune)
##
##               Inertia Rank
## Total           84.12
## Unconstrained   84.12   19
## Inertia is variance
##
## Eigenvalues for unconstrained axes:
##    PC1    PC2    PC3    PC4    PC5    PC6    PC7    PC8
## 24.795 18.147  7.629  7.153  5.695  4.333  3.199  2.782
## (Showed only 8 of all 19 unconstrained eigenvalues)
```

```
plot(pca.dune)
```



Mamy rozrzut poletek i gatunków. Dwie pierwsze osie wyjaśniają odpowiednio 24.795/84.12 = 29,5% oraz 18.147/84.12=21,6% zmienności, czyli całkiem nieźle. Wiemy jak różnią się między sobą poletka, pytanie co może na to wpływać. Mamy dane o zawartości aluminium w glebie i wilgotności. Spróbujmy dodać je do wyniku analizy. Możemy zrobić to na dwa sposoby - za pomocą projekcji pasywnej oraz w sposób aktywny - zmieniający rozrzut punktów w przestrzeni. W sposób pasywny można dodać wektory do każdej analizy za pomocą funkcji `envfit()` z dwoma argumentami - pierwszym jest wynik analizy, drugim - data.frame z czynnikami do dodania:

```
data(dune.env)
summary(dune.env)
```

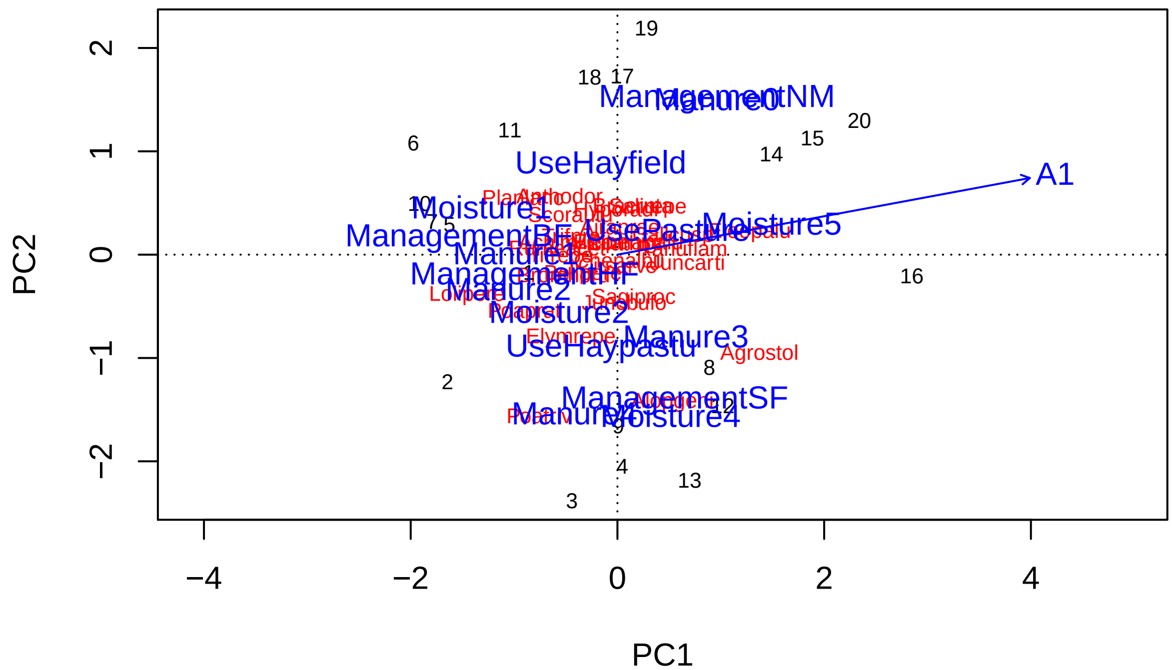
```
##        A1          Moisture Management       Use     Manure
##  Min.   : 2.800   1:7      BF:3       Hayfield:7   0:6
##  1st Qu.: 3.500   2:4      HF:5       Haypastu:8   1:3
##  Median : 4.200   4:2      NM:6       Pasture :5   2:4
##  Mean   : 4.850   5:7      SF:6                    3:4
##  3rd Qu.: 5.725                                    4:3
##  Max.   :11.500
```

```
pasywnie<-envfit(pca.dune, dune.env)
pasywnie
```

```
##
## ***VECTORS
##
##        PC1     PC2     r2 Pr(>r)
## A1 0.98316 0.18274 0.2632  0.045 *
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Permutation: free
## Number of permutations: 999
##
## ***FACTORS:
##
## Centroids:
##                   PC1     PC2
## Moisture1    -1.3148  0.4591
## Moisture2    -0.5650 -0.5518
## Moisture4     0.5136 -1.5584
## Moisture5     1.4909  0.3015
## ManagementBF -1.5335  0.1574
## ManagementHF -0.8985 -0.2111
## ManagementNM  0.9626  1.5060
## ManagementSF  0.5529 -1.4088
## UseHayfield  -0.1616  0.8657
## UseHaypastu  -0.1580 -0.9086
## UsePasture    0.4790  0.2417
## Manure0       0.9626  1.5060
## Manure1      -0.9822  0.0150
## Manure2      -1.0563 -0.3299
## Manure3       0.6613 -0.7901
## Manure4      -0.4163 -1.5338
##
## Goodness of fit:
##                 r2 Pr(>r)
## Moisture   0.4709  0.001 ***
## Management 0.5540  0.001 ***
## Use        0.1710  0.171
## Manure     0.4851  0.004 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Permutation: free
## Number of permutations: 999
```

```r
plot(pca.dune)
plot(pasywnie, add=T)
```



W ten sposób nanosząc kolorem niebieskim czynniki widzimy które poletka (numery czarnym kolorem) są związane z danym poziomem zmiennych jakościowych (kolorem niebieskim). Możemy też wnioskować o preferencjach gatunków (czerwonym). Wartość A1 została przedstawiona jako zmienna ilościowa - widzimy relacje między rozrzutem poletek o różnym poziomie A1 i odpowiedzi gatunków. Wywołanie obiektu `pasywnie` zwróciło nam podsumowanie dopasowania - dla zmiennych - widzimy test dopasowania w oparciu o 999 permutacji i współczynnik determinacji.

Dopasowanie aktywne, czyli zastosowanie RDA, zmieni układ punktów w PCA. Do wykonania tej analizy użyjemy podobnie jak w przypadku PCA funkcji `rda`, lecz zapisaną inaczej. Pierwszym argumentem będzie formuła, gdzie zmienną zależną będzie `data.frame` z danymi, a niezależnymi - nazwy kolumn z drugiego argumentu:

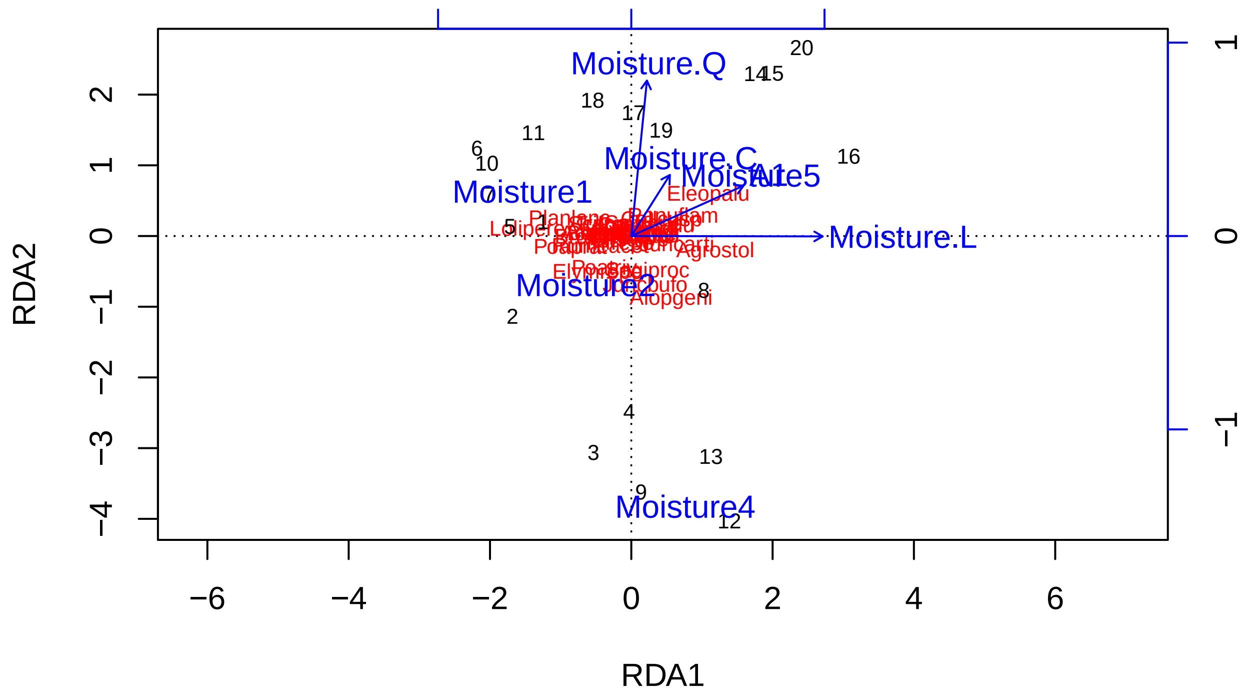
```
rda1<-rda(dune~A1+Moisture, data=dune.env)
rda1
```

```
## Call: rda(formula = dune ~ A1 + Moisture, data = dune.env)
##
##               Inertia Proportion Rank
## Total         84.1237     1.0000
## Constrained   29.7645     0.3538    4
## Unconstrained 54.3592     0.6462   15
## Inertia is variance
##
## Eigenvalues for constrained axes:
##   RDA1   RDA2   RDA3   RDA4
## 19.333  5.850  2.905  1.676
##
## Eigenvalues for unconstrained axes:
```

```
##    PC1    PC2    PC3    PC4    PC5    PC6    PC7    PC8    PC9   PC10
## 15.027  9.420  6.222  5.137  4.928  3.222  3.036  2.269  1.657  1.544
##   PC11   PC12   PC13   PC14   PC15
##  0.733  0.407  0.345  0.235  0.176
```

W podsumowaniu mamy w przeciwieństwie do PCA rozdzieloną inercję układu - na czynniki ograniczone (*Constrained*) i nieograniczone (*Unconstrained*) - widzimy że dodane czynniki wyjaśniają 35,38% zmienności układu. Osie PC1 i PC2 wyjaśniają odpowiednio 15.027/54.3592 = 27,6% oraz 9.420/54.3592 =17,3%. Mamy więc wyjaśnione więcej zmienności niż w przypadku prostej PCA. Jak wygląda wynik - podobnie:

```r
plot(rda1)
```



CCA - Canonical Correspondence Analysis

W przypadku CCA sytuacja jest analogiczna do RDA, z tym że zamiast PCA analizą bazową jest CA. Używamy funkcji `cca()` z formułą wskazaną za pomocą identycznej składni co w przypadku RDA.

3. Przygotowanie danych, transformacje

Omawiane analizy wymagają danych w postaci `data.frame`, w której cechy są kolumnami a obiekty - wierszami. Wartości NA nie są dozwolone. Z tego względu w trzech użytych niżej przykładach imputowano wartości za pomocą analizy random forest z pakietu `missForest`. Pakiet `vegan` zawiera funkcję pozwalającą na preprocessing danych - `decostand()`. Najczęściej do preprocessingu używa się następujących metod:

- `total` - wartości są dzielone przez sumę wartości danej cechy
- `freq` - wartości są dzielone przez maksimum danej cechy i mnożone przez liczbę niezerowych obserwacji
- `normalize` - transformacja polegająca na wystandaryzowaniu sumy kwadratów każdej z cech do wartości 1
- `range` - wystandaryzowanie cech w zakresie od 0 do 1
- `standardize` - skalowanie w taki sposób, by średnia wynosiła 0, a wariancja - 1
- `log` - transformacja typu `log1p()`

4. Przypadek 1 - czym się różnią drzewa?

zbiór danych

```
drzewa<-read.csv('trees.csv',sep=';')
summary(drzewa)
```

```
##              nazwa                       spec        alien
##  brzoza br.     : 1   Abies alba         : 1   obcy   :19
##  brzoza om.     : 1   Acer campestre     : 1   rodzimy:35
##  buk            : 1   Acer negundo       : 1
##  choina         : 1   Acer platanoides   : 1
##  cis            : 1   Acer pseudoplatanus: 1
##  czeremcha amer.: 1   Acer saccharinum   : 1
##  (Other)        :48   (Other)            :48
##     wdensity           lai         canopy_height      leaf_dmc
##  Min.   :0.2840   Min.   : 2.773   Min.   : 3.167   Min.   :206.2
##  1st Qu.:0.4177   1st Qu.: 4.005   1st Qu.:14.875   1st Qu.:265.9
##  Median :0.4957   Median : 4.656   Median :22.015   Median :280.6
##  Mean   :0.4911   Mean   : 4.947   Mean   :24.090   Mean   :280.8
##  3rd Qu.:0.5596   3rd Qu.: 5.333   3rd Qu.:31.641   3rd Qu.:298.5
##  Max.   :0.7000   Max.   :12.642   Max.   :65.000   Max.   :369.8
##
##    leaf_mass         leaf_size      seed_mass              pochodzenie
##  Min.   :  5.395   Min.   :   0   Min.   :   0.100   East Asia    : 1
##  1st Qu.:102.753   1st Qu.:1529   1st Qu.:   7.679   Europe       :41
##  Median :151.885   Median :3061   Median :  63.811   North America:12
##  Mean   :141.144   Mean   :3023   Mean   : 636.140
##  3rd Qu.:179.205   3rd Qu.:3914   3rd Qu.: 309.079
##  Max.   :294.430   Max.   :8365   Max.   :7708.650
##
##       iglasty   shade_tolerance drought_tolerance waterlogging_tolerance
##  iglasty  :12   Min.   :1.350   Min.   :1.000     Min.   :1.000
##  lisciasty:42   1st Qu.:2.277   1st Qu.:2.223     1st Qu.:1.105
##                 Median :2.765   Median :2.740     Median :1.390
##                 Mean   :2.931   Mean   :2.731     Mean   :1.798
##                 3rd Qu.:3.515   3rd Qu.:3.010     3rd Qu.:2.060
##                 Max.   :4.830   Max.   :4.470     Max.   :4.100
##
##  disturbance_index
##  Min.   :-1.9910
##  1st Qu.:-1.8758
##  Median :-1.1132
##  Mean   :-1.0452
##  3rd Qu.:-0.1362
##  Max.   : 0.0000
##
```

Zbiór danych zawiera informacje o cechach gatunków drzew. Zebrano następujące informacje: nazwa - nazwa polska spec - nazwa naukowa alien - czy gatunek obcy, czy rodzimy - factor wdensity – gęstość drewna [t/m3] lai – indeks powierzchni liści [m2 liści/ m2 glebyy] canopy_height – wysokość [m] leaf_dmc – zawartość suchej masy w liściu [mg/g] leaf_mass – masa liści [mg] leaf_size – powierzchnia [cm2] seed_mass – masa nasion [mg] pochodzenie - kontynent [factor] iglasty - czy gatunek jest iglasty, czy liściasty [factor] shade_tolerance - wskaźnik cienioznośności drought_tolerance - wskaźnik odporności na suszę waterlogging_tollerance –

wskaźnik odporności na zalewanie, te trzy wskaźniki mają wartości od 0 do 5 disturbance_index – wskaźnik odporności na zaburenia, dla drzew ma wartości od -2 do 0

Mamy zmienne ilościowe i kilka jakościowych. Do analiz należy wziąć jedynie zmienne ilościowe - wykluczmy więc także zmienne jakościowe zakodowane jako 0/1:

```
drzewa.an<-drzewa[,-c(1,2,3,11,12)]
```

wybór metody

W tym zbiorze danych nie ma dodatkowych zmiennych które mogłyby nam pomóc wyjaśnić zmienność, więc wykonamy tylko analizy nieograniczone. Pozostaje pytanie o długość gradientów. Zanim jednak wykonamy DCA dobrze byłoby zrobić preprocessing danych. Sprawdźmy który z nich będzie się lepiej modelował za pomocą DCA:

```
decorana(decostand(drzewa.an, method='range'))
```

```
##
## Call:
## decorana(veg = decostand(drzewa.an, method = "range"))
##
## Detrended correspondence analysis with 26 segments.
## Rescaling of axes with 4 iterations.
##
##                    DCA1    DCA2    DCA3    DCA4
## Eigenvalues     0.1082 0.10742 0.06731 0.03447
## Decorana values 0.1153 0.08706 0.05288 0.02570
## Axis lengths    1.5314 1.58143 1.26278 0.90602
```

```
decorana(decostand(drzewa.an, method='freq'))
```

```
## Warning in decostand(drzewa.an, method = "freq"): input data contains negative entries: result may be

## Warning in decorana(decostand(drzewa.an, method = "freq")): some species
## were removed because they were missing in the data

##
## Call:
## decorana(veg = decostand(drzewa.an, method = "freq"))
##
## Detrended correspondence analysis with 26 segments.
## Rescaling of axes with 4 iterations.
##
##                    DCA1    DCA2    DCA3     DCA4
## Eigenvalues     0.1730 0.05337 0.01714 0.017882
## Decorana values 0.2823 0.03902 0.01434 0.007638
## Axis lengths    1.6117 0.99521 0.60484 0.529280
```

Metoda `'freq'` daje lepsze wyniki. Z uwagi na to, że wykonujemy DCA niektóre metody preprocessingy - np. `'standardize'` czy `'normalize'` nie mogą zostać zastosowane. Widzimy że długość gradientu jest niewielka - możemy zastosować CA lub PCA. Sprawdźmy różne standardyzacje danych i obie analizy:

```
rda(decostand(drzewa.an, method='standardize'))
```

```
## Call: rda(X = decostand(drzewa.an, method = "standardize"))
##
##              Inertia Rank
## Total             11
```

```
## Unconstrained      11   11
## Inertia is variance
##
## Eigenvalues for unconstrained axes:
##    PC1    PC2    PC3    PC4    PC5    PC6    PC7    PC8    PC9   PC10
## 3.0220 1.9878 1.2726 1.0127 0.8330 0.7704 0.6294 0.5159 0.4329 0.3444
##   PC11
## 0.1790
```

```r
rda(decostand(drzewa.an, method='normalize'))
```

```
## Call: rda(X = decostand(drzewa.an, method = "normalize"))
##
##               Inertia Rank
## Total          0.1892
## Unconstrained  0.1892   11
## Inertia is variance
##
## Eigenvalues for unconstrained axes:
##     PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9
## 0.12956 0.05462 0.00272 0.00223 0.00011 0.00000 0.00000 0.00000 0.00000
##    PC10    PC11
## 0.00000 0.00000
```

```r
cca(decostand(drzewa.an, method='range'))
```

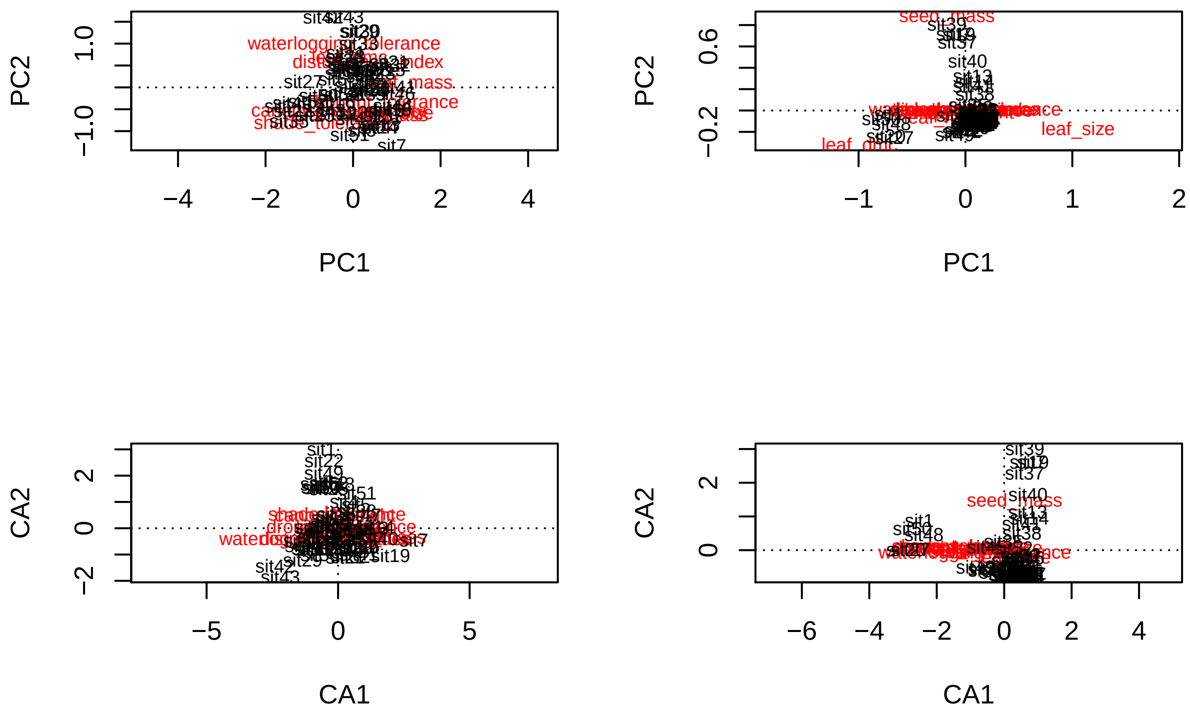
```
## Call: cca(X = decostand(drzewa.an, method = "range"))
##
##               Inertia Rank
## Total          0.4563
## Unconstrained  0.4563   10
## Inertia is mean squared contingency coefficient
##
## Eigenvalues for unconstrained axes:
##     CA1     CA2     CA3     CA4     CA5     CA6     CA7     CA8     CA9
## 0.11528 0.10835 0.06631 0.05755 0.03441 0.02198 0.02007 0.01639 0.01107
##    CA10
## 0.00487
```

```r
cca(decostand(drzewa.an, method='normalize'))
```

```
## Call: cca(X = decostand(drzewa.an, method = "normalize"))
##
##               Inertia Rank
## Total          0.6114
## Unconstrained  0.6114    9
## Inertia is mean squared contingency coefficient
## 1 species (variable) deleted due to missingness
##
## Eigenvalues for unconstrained axes:
##    CA1    CA2    CA3    CA4    CA5    CA6    CA7    CA8    CA9
## 0.3273 0.2546 0.0251 0.0033 0.0007 0.0002 0.0002 0.0001 0.0000
```

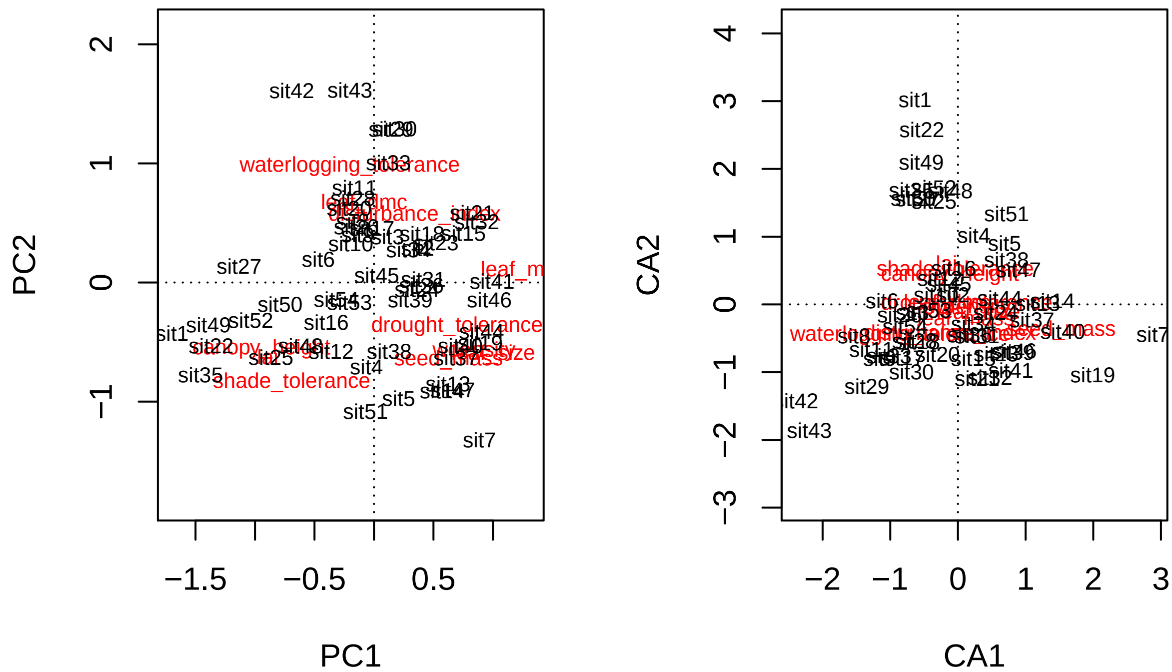
Widzimy że mamy analizy w których pierwszy komponent wyjaśnia większą część zmienności, i mamy takie gdzie niekoniecznie. Ciężko stwierdzić co lepsze - więcej zmienności czy jakość wyników. Spójrzmy jak wyglądają wyniki:

```r
par(mfrow=c(2,2))
plot(rda(decostand(drzewa.an, method='standardize')))
plot(rda(decostand(drzewa.an, method='normalize')))
plot(cca(decostand(drzewa.an, method='range')))
plot(cca(decostand(drzewa.an, method='normalize')))
```



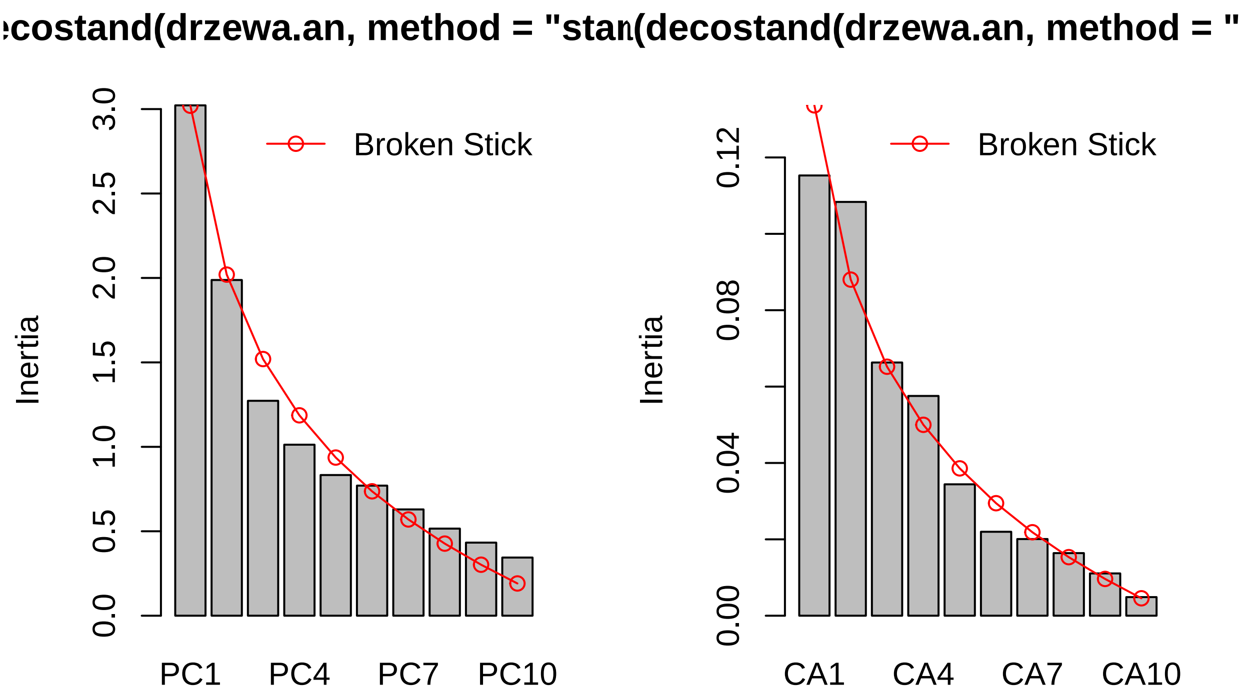
Widzimy różny kształt rozrzutu punktów. W dwóch przypadkach widzimy upakowanie punktów. Teoretycznie taki przypadek powinniśmy wykluczyć, jednak tej standaryzacji danych nie mogliśmy sprawdzić - `decorana()` nie obsługuje wartości ujemnych. Widzimy jednak na wykresach że to sztuczne upakowanie punktów jest artefaktem. Po odrzuceniu tej metody zostają nam dwa wykresy po lewej stronie. Przyjrzyjmy się im bliżej:

```r
par(mfrow=c(1,2))
plot(rda(decostand(drzewa.an, method='standardize')))
plot(cca(decostand(drzewa.an, method='range')))
```



W pierwszym przypadku - PCA - punkty reprezentujące obiekty tworzą romb, w drugim - CA - trójkąt. Powstaje pytanie - jak ocenić jakość dopasowania. Można posłużyć się ilością zmienności wyjaśnianej przez osie, można zastosować wykres diagnostyczny z funkcji `screeplot()` - pokazuje on rozkład zmienności wyjaśnianej przez poszczególne osie analizy. Argument `bstick=T` dodaje do wykresów model złamanego patyka - teoretyczny rozkład inercji dla pełnej losowości. Ten test może być kryterium wyboru liczby osi do interpretacji - zgodnie z regułą kciuka powinno się wybierać osie do momentu w którym inercja przestanie być większa niż w przypadku modelu teoretycznego:

```
par(mfrow=c(1,2))
screeplot(rda(decostand(drzewa.an, method='standardize')), bstick=T)
screeplot(cca(decostand(drzewa.an, method='range')),bstick=T)
```

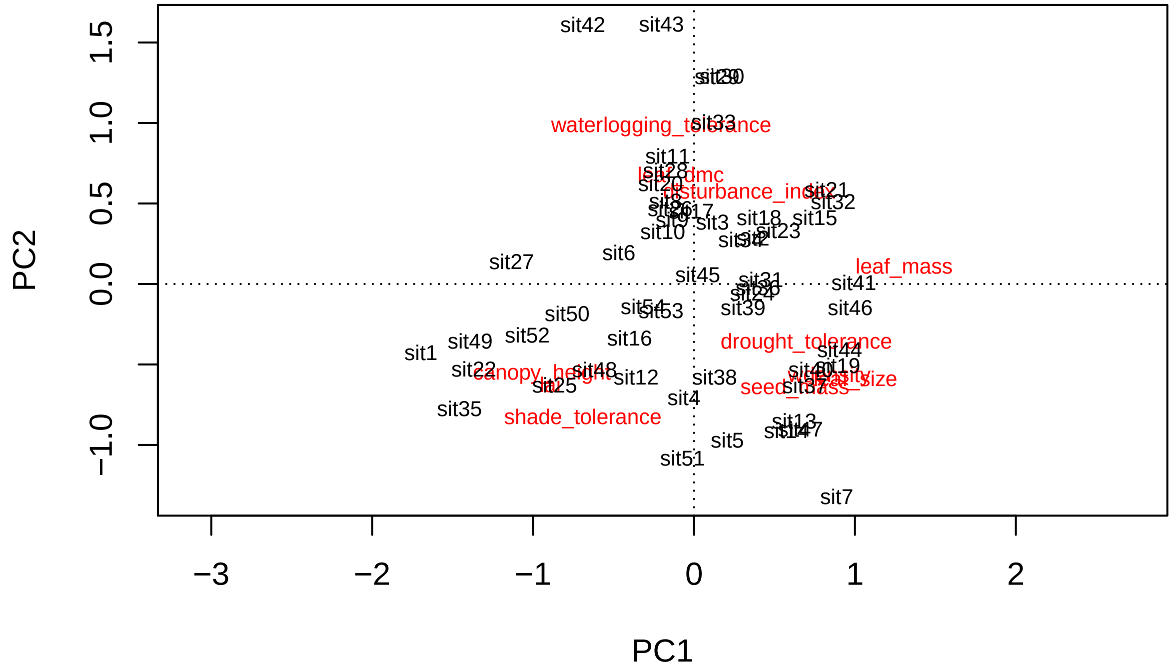


W naszym przypadku dla PCA możemy wziąć pod uwagę oś 1, oś 2 jest tuż pod granicą modelu, w przypadku CA oś 1 jest pod modelem teoretycznym. Niemniej jednak zawsze analizowane będą co najmniej dwie osie. Możliwa jest także sytuacja gdy pierwsza oś nie będzie dawać znaczących wyników natomiast oś druga i trzecia pozwoli na lepsze wnioskowanie. Z puktu widzenia interpretacji rozrzut punktów w formie trójkąta jest niekorzystny - świadczy o braku związku pomiędzy znalezionymi gradientami a analizowanymi cechami. W związku z tym najlepszym wyborem będzie PCA.

wizualizacja

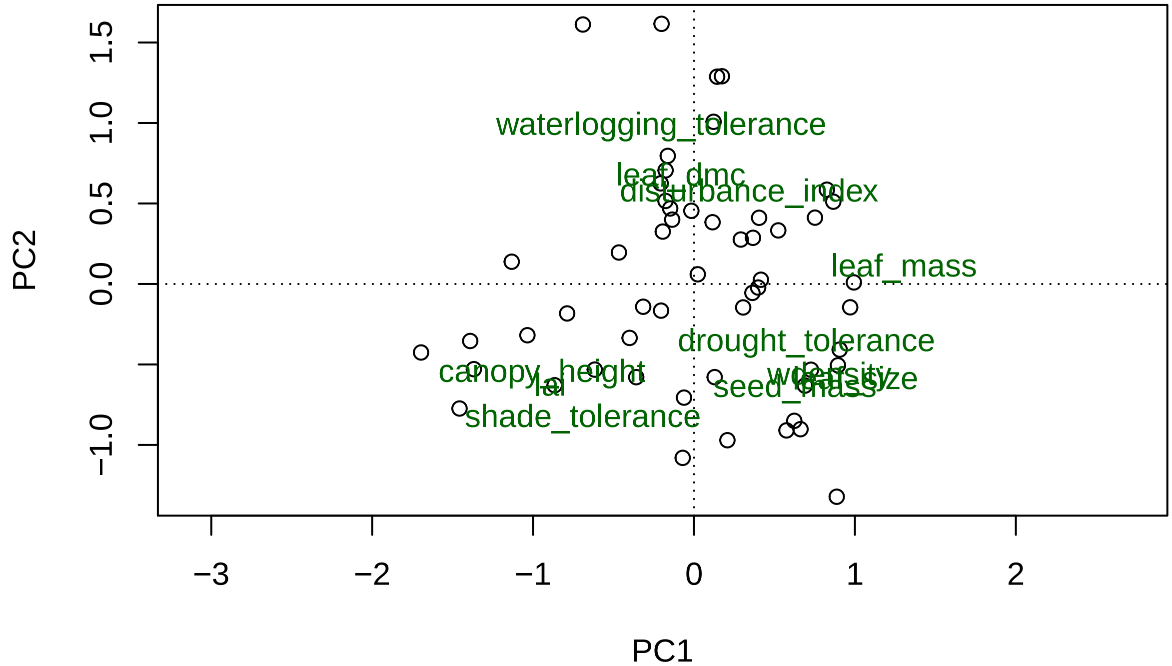
Po wyborze metody możemy zająć się wizualizacją danych. Pakiet `vegan` oferuje kilka rozwiązań. Pierwsze z nich to po prostu przeciążona funkcja `plot()`:

```
par(mfrow=c(1,1))
tree.pca<-rda(decostand(drzewa.an, method='standardize'))
plot(tree.pca)
```



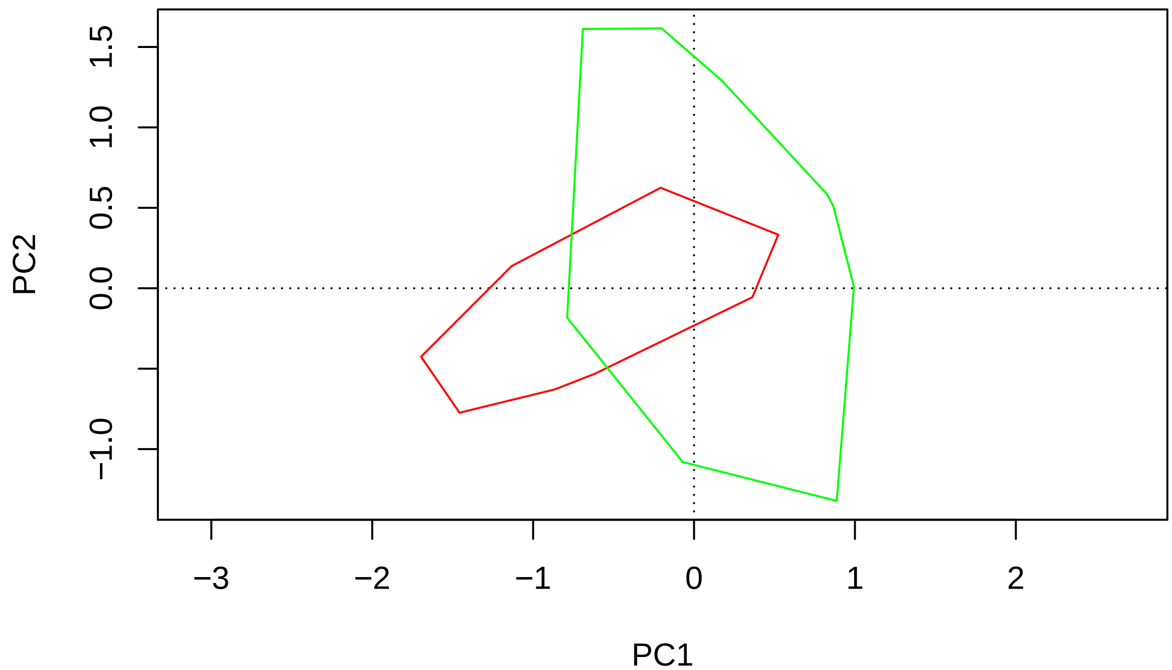
Drugą opcją jest narysowanie najpierw pustej ramy, a potem wypełnienie jej punktami/etykietami za pomocą funkcji `points()` i `text()`:

```
plot(tree.pca, type='none')
points(tree.pca, display='sites', pch=1)
text(tree.pca, display='species', col='darkgreen')
```



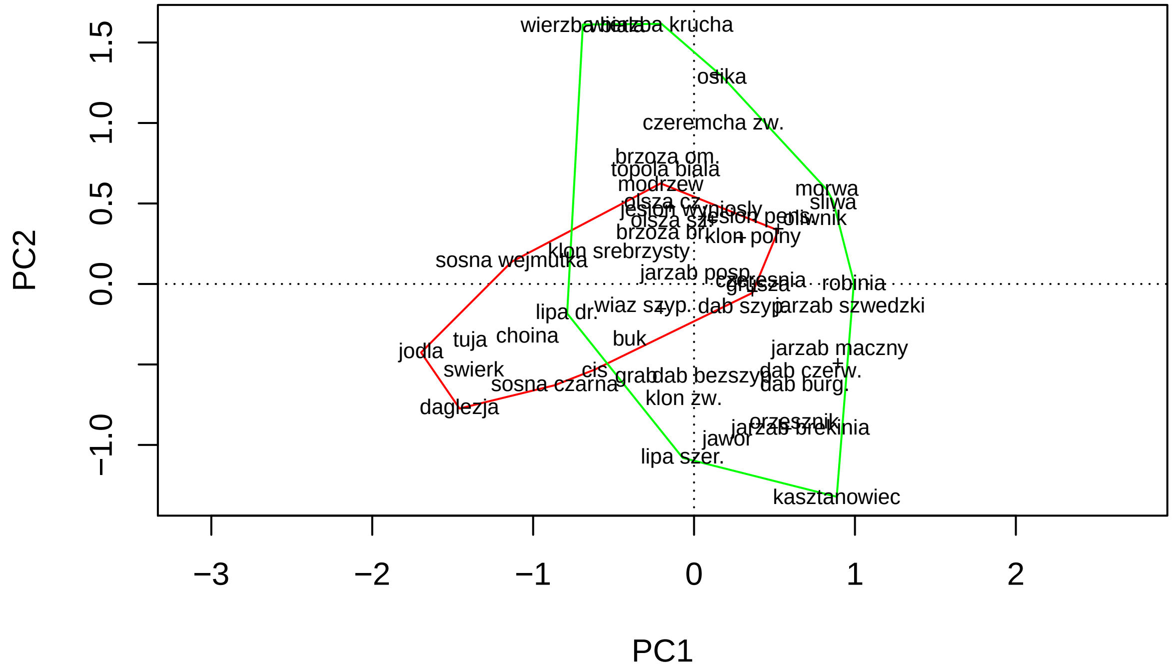
Pamiętamy, że `sites` to obiekty a `species` to cechy. Możemy też sprawdzić jak grupują się nasze obiekty za pomocą funkcji `ordihull()`, w której podajemy nasz wynik analizy, wektor z nazwami grup (tutaj kolumnę `iglasty` będącą factorem z dwoma poziomami) oraz wektor kolorόW:

```
plot(tree.pca, type='none')
ordihull(tree.pca, groups =drzewa$iglasty,col=c('red','green'))
```



Widzmy, że mamy dwie grupy, częściowo rozłączne. Nie wiemy natomiast które gatunki są które. Dołóżmy do tego etykiety obiektów. Aby nie zaciemniać wykresu dołóżmy je jednak za pomocą funkcji `orditorp()` - wstawiającej etykiety wg wektora priorytetów. W naszym przypadku jako wektor etykiet (argument `label`) podamy kolumnę `spec` z nazwami gatunków, a jako wektor priorytetów - gęstość drewna - `wdensity`. Parametr `air` określa odstęp pomiędzy etykietami:

```
plot(tree.pca, type='none')
ordihull(tree.pca, groups =drzewa$iglasty,col=c('red','green'))
orditorp(tree.pca,display='sites',label=as.character(drzewa$nazwa),priority=drzewa$wdensity,pch='+',air
```



ggplot

Mimo wszystko nie jesteśmy zadowoleni z tych obrazków. Chcielibyśmy móc łatwiej sterować grafiką i żeby to lepiej wyglądało. Aż się prosi o ggploty, ale nasze obiekty nie są data frame'ami. Trzeba wyekstrachować więc punkty:

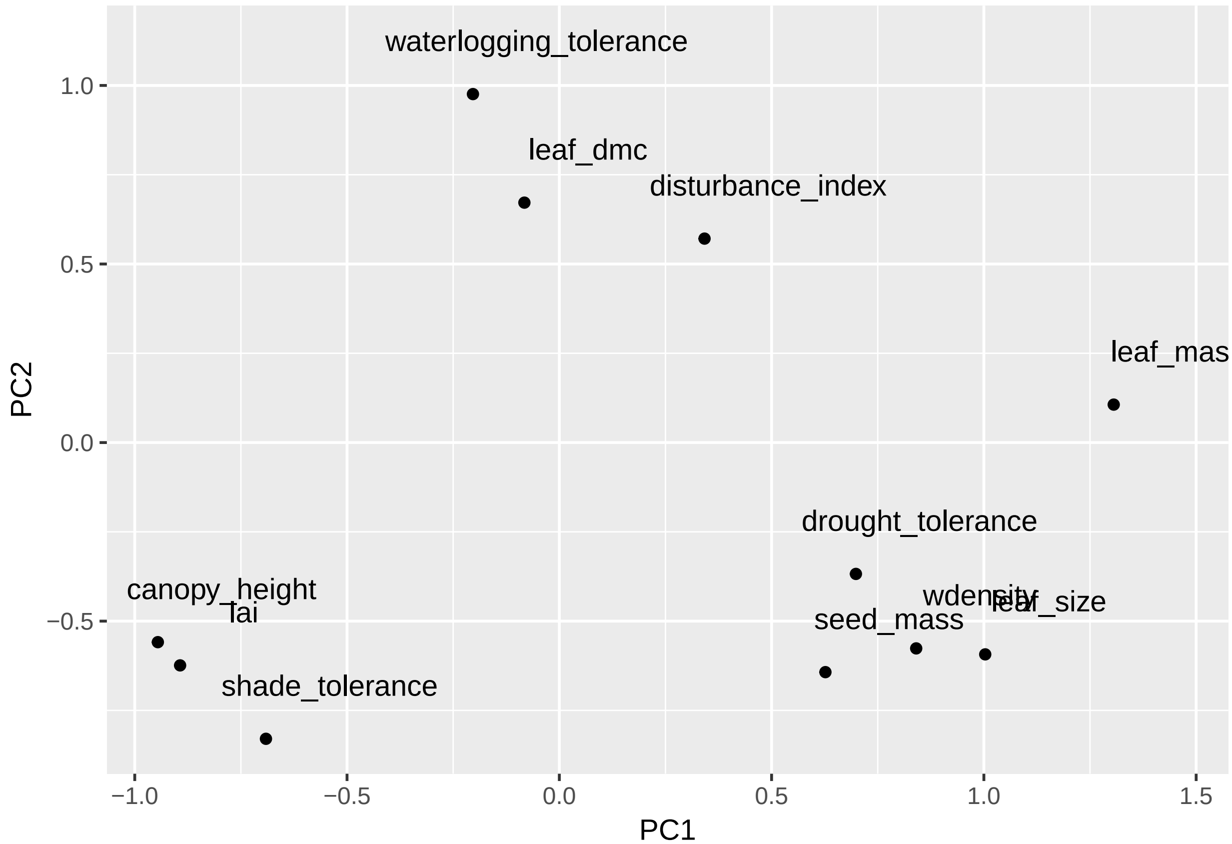
```
tree.gg.obiekty<-as.data.frame(scores(tree.pca,display = 'sites'))
tree.gg.cechy<-as.data.frame(scores(tree.pca, display='species'))
tree.gg.cechy$name<-rownames(tree.gg.cechy)
```

do obiektów dostawmy kolumny z nazwami gatunków i informacjami zapisanymi jako factory:

```
tree.gg.obiekty<-cbind(tree.gg.obiekty, drzewa[,c(1,2,3,11,12)])
```

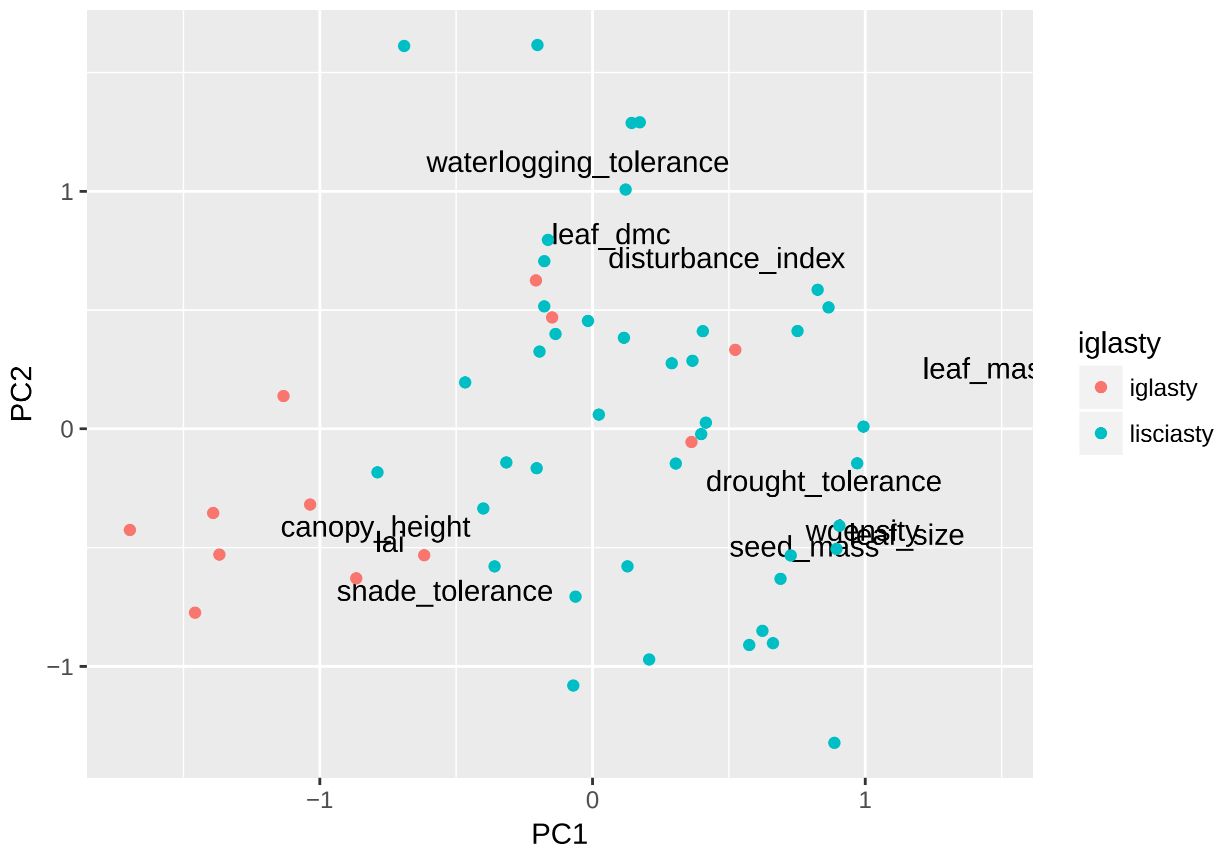
spróbujmy to zwizualizować

```
library(ggplot2)
g<-ggplot(tree.gg.cechy, aes(x=PC1,y=PC2))+geom_point()
g+geom_text(aes(label=name),nudge_x = .15,nudge_y = .15)
```



Na pierwszy ogień poszły cechy. Mamy trzy główne kierunki zmienności. Oś PC1 różnicuje nam drzewa na gatunki o dużej wysokości, indeksie powierzchni liściowej i tolerancji na zacienienie po lewej stronie i gatunkach o dużej gęstości drewna, masie nasion i rozmiarach liści po prawej. Wzdłuż osi PC2 rośnie odporność na zalewanie, zawartość suchej masy w liściu i odporność na zaburzenia. Widzimy że waterlogging tolerance i drought tolerance są przeciwstawne, co jest dość logiczne. Wysokość drzewa jest największa tam, gdzie mniejsza odporność na suszę czy zalewanie, a tam gdzie cienioznośność. Zobaczmy co będzie gdy dodamy do tego obrazu nasze obiekty:

```
g<-ggplot(tree.gg.cechy, aes(x=PC1,y=PC2))
g<-g+geom_text(aes(label=name),nudge_x = .15,nudge_y = .15)
g+geom_point(data=tree.gg.obiekty,aes(col=iglasty))
```



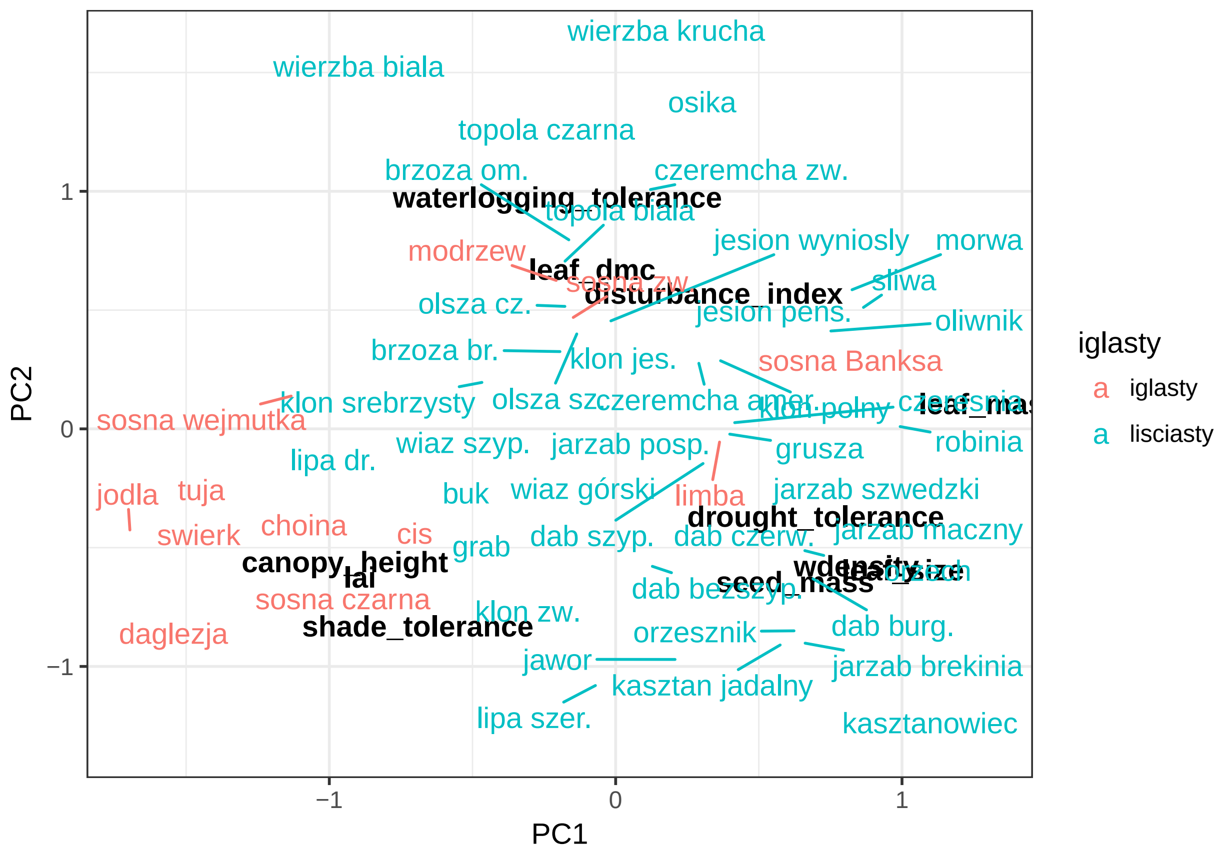
Widzimy że po lewej stronie mamy gatunki iglaste, po prawej zaś - liściaste. Drewno gatunków liściastych jest cięższe i z reguły są to gatunki o większych nasionach (małe nasiona sosen ukryte w szyszkach vs. żołędzie czy kasztany). Masa i rozmiar igieł też będą z reguły mniejsze niż liści. LAI może być jednak większe, bo jest to powierzchnia liści nad metrem kwadratowym gleby, a u gatunków iglastych poziom "upakowania" igliwia w pionie jest duży.

Co się stanie jeśli dodamy nazwy drzew?

```
g<-ggplot(tree.gg.cechy, aes(x=PC1,y=PC2))
g<-g+geom_text(aes(label=name), col='black',fontface='bold')
g<-g+geom_text(data=tree.gg.obiekty,
               aes(col=iglasty,label=nazwa),position=position_jitter(width=.1,height=.1))
```

Prawie mamy coś co jakoś wygląda. Problemem są nachodzące na siebie nazwy. Zróbmy coś z tym za pomocą pakietu `ggrepel` i funkcji `geom_text_repel()`, która ma taką samą składnie jak `geom_text()`:

```
library(ggrepel)
g<-ggplot(tree.gg.cechy, aes(x=PC1,y=PC2))
g<-g+geom_text(aes(label=name), col='black',fontface='bold')
g<-g+geom_text_repel(data=tree.gg.obiekty,aes(col=iglasty,label=nazwa))
g+theme_bw()
```



W kierunku wskazywanym przez odporność na zalewanie położone są wierzby i topole - gatunki nadrzeczne. W odwrotnym - grab, buk, klon i dąb bezszypułkowy - gatunki raczej nie związane z wodą. Przy seed mass mamy jarzęby, kasztana, orzechy i inne duże nasiona. Dęby są skierowane w stronę wdensity. Cztery gatunki iglaste odstające od lewej strony to modrzew, sosna zwyczajna, s. Banksa i limba. Wolą tereny otwarte, są bardziej światłożądne, tolerują zaburzenia.

Jak jeszcze można zadbać o estetykę wykresu? Możemy zmniejszyć czcionkę, ale tracimy na czytelności napisów. Jeśli naszym celem jest wąska grupa osób znających obiekty to warto zastąpić nazwy skrótami, np. czteroliterowymi. Do nazw łacińskich `vegan` ma funkcję `makecepnames()` która tworzy skróty składające się z czterech pierwszych liter nazwy rodzajowej i gatunkowej. Można też wybrać tylko kilka skrajnych nazw do pokazania, prezentujących główne trendy. Opcji eksperymentowania jest wiele.

5. Przypadek 2. Co wpływa na naszą ocenę piwa?

zbiór danych

```
piwa<-read.csv('beer.csv',sep=';')
summary(piwa)
```

```
##                               browar                          nazwa
##  AleBrowar                       :12    Łódź i Młyn              :  1
##  Brokreacja                      :12    Pacific                  :  1
##  Pinta                           :11   10.5                      :  1
##  Browar Setka                    :10   55 Porter Bałtycki Wędzony :  1
##  Browar Szałpiw                  :10   652 n.p.m.                :  1
##  JAN OLBRACHT BROWAR RZEMIEŚLNICZY:10   A Ja Pale Ale             :  1
##  (Other)                         :60   (Other)                   :119
##                 styl          BLG              V              gat_jedn
##  Pale Lager         :  6   Min.   : 8.00   Min.   : 3.000   pale ale :37
##  Strong Lager       :  5   1st Qu.:12.00   1st Qu.: 5.200   eurolager:13
##   Witbier           :  4   Median :14.00   Median : 6.000   inne     :13
##   Pszeniczne        :  3   Mean   :15.07   Mean   : 6.301   Wheat    :13
##   RIS               :  3   3rd Qu.:16.00   3rd Qu.: 6.725   Stout    :10
##   American PAle Ale :  2   Max.   :32.00   Max.   :13.800   Porter   : 9
##  (Other)            :102                                    (Other)  :30
##     overall           style             IBU            weighted
##  Min.   :  2.00   Min.   :  8.00   Min.   :  7.00   Min.   :1.670
##  1st Qu.: 45.00   1st Qu.: 37.00   1st Qu.: 30.00   1st Qu.:3.130
##  Median : 73.00   Median : 78.00   Median : 40.00   Median :3.370
##  Mean   : 64.82   Mean   : 64.77   Mean   : 44.08   Mean   :3.283
##  3rd Qu.: 92.00   3rd Qu.: 94.00   3rd Qu.: 57.30   3rd Qu.:3.610
##  Max.   :100.00   Max.   :100.00   Max.   :120.00   Max.   :4.240
##
```

Zbiór danych zawiera informacje o cechach 125 piw i ich ocenach na ratebeer.com. Mamy zmienne ilościowe i jakościowe - w tym style piwne zapisane w wersji ogólniejszej i szczegółowej. Zróżnicowanie jest olbrzymie, stąd pewnie będzie nas bardziej interesowało uchwycenie zmienności w ramach głównych kategorii.

```
piwa.an<-piwa[,-c(1,2,3,6)]
rownames(piwa.an)<-piwa$nazwa
```

wybór metody

W tym zbiorze danych mamy niewiele cech - zaledwie sześć, z czego tylko trzy opisują właściwości piwa: BLG (procent ekstraktu), V (zawartość etanolu) oraz IBU (zawartość alfa-kwasów chmielowych, czyli miara goryczki; International Bitternesss Unit). Pozostałe trzy to oceny użytkowników ratebeer.com: weighted to ocena w skali 0-5, natomiast style oraz overall to percentyle w których dane piwo się znajduje w skali całego portalu oraz w ramach danego stylu piwnego. To zróżnicowanie jest związane z różną oceną różnych styli - z reguły lagery dostają niskie oceny, a "wymyślne" piwa typu tripel, quadrupel czy barley wine są przeceniane. Ze względu na różne skale musimy znów zastanowić się nad preprocessingiem. Normalize odpada bo zaniży część wartości. Sprawdźmy jak długi jest gradient w tych danych za pomocą DCA.

```r
decorana(decostand(piwa.an, method='range'))
```

```
##
## Call:
## decorana(veg = decostand(piwa.an, method = "range"))
##
## Detrended correspondence analysis with 26 segments.
## Rescaling of axes with 4 iterations.
##
##                     DCA1    DCA2    DCA3     DCA4
## Eigenvalues     0.05693 0.03656 0.02269 0.011292
## Decorana values 0.05717 0.03417 0.01863 0.007883
## Axis lengths    1.60733 1.04806 1.06177 0.621413
```

```r
decorana(decostand(piwa.an, method='normalize'))
```

```
##
## Call:
## decorana(veg = decostand(piwa.an, method = "normalize"))
##
## Detrended correspondence analysis with 26 segments.
## Rescaling of axes with 4 iterations.
##
##                     DCA1   DCA2    DCA3     DCA4
## Eigenvalues     0.07633 0.0359 0.01521 0.019709
## Decorana values 0.07634 0.0326 0.00938 0.002855
## Axis lengths    1.15753 0.9355 0.59128 0.656415
```

Znów oba przypadku wskazują na krótkie gradienty. Sprawdźmy CA i PCA:

```r
rda(decostand(piwa.an, method='freq'))
```

```
## Call: rda(X = decostand(piwa.an, method = "freq"))
##
##                Inertia Rank
## Total           0.8969
## Unconstrained   0.8969    6
## Inertia is variance
##
## Eigenvalues for unconstrained axes:
##    PC1    PC2    PC3    PC4    PC5    PC6
## 0.5239 0.2027 0.0848 0.0526 0.0314 0.0016
```

```r
rda(decostand(piwa.an, method='range'))
```

```
## Call: rda(X = decostand(piwa.an, method = "range"))
##
##                Inertia Rank
## Total           0.3494
## Unconstrained   0.3494    6
## Inertia is variance
##
## Eigenvalues for unconstrained axes:
##     PC1     PC2     PC3     PC4     PC5     PC6
## 0.23139 0.05412 0.02839 0.02357 0.00990 0.00203
```

```r
cca(decostand(piwa.an, method='range'))
```

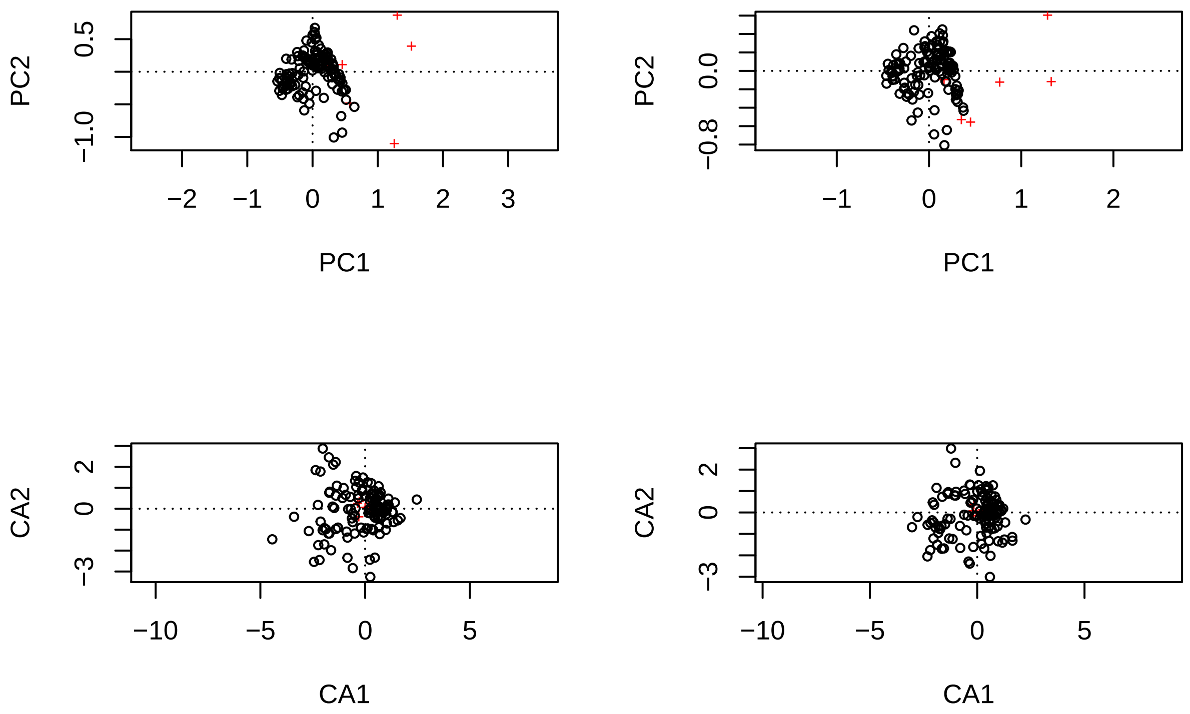
```
## Call: cca(X = decostand(piwa.an, method = "range"))
##
##               Inertia Rank
## Total          0.1306
## Unconstrained  0.1306    5
## Inertia is mean squared contingency coefficient
##
## Eigenvalues for unconstrained axes:
##     CA1     CA2     CA3     CA4     CA5
## 0.05717 0.03605 0.02183 0.01032 0.00525
```

```r
cca(decostand(piwa.an, method='freq'))
```

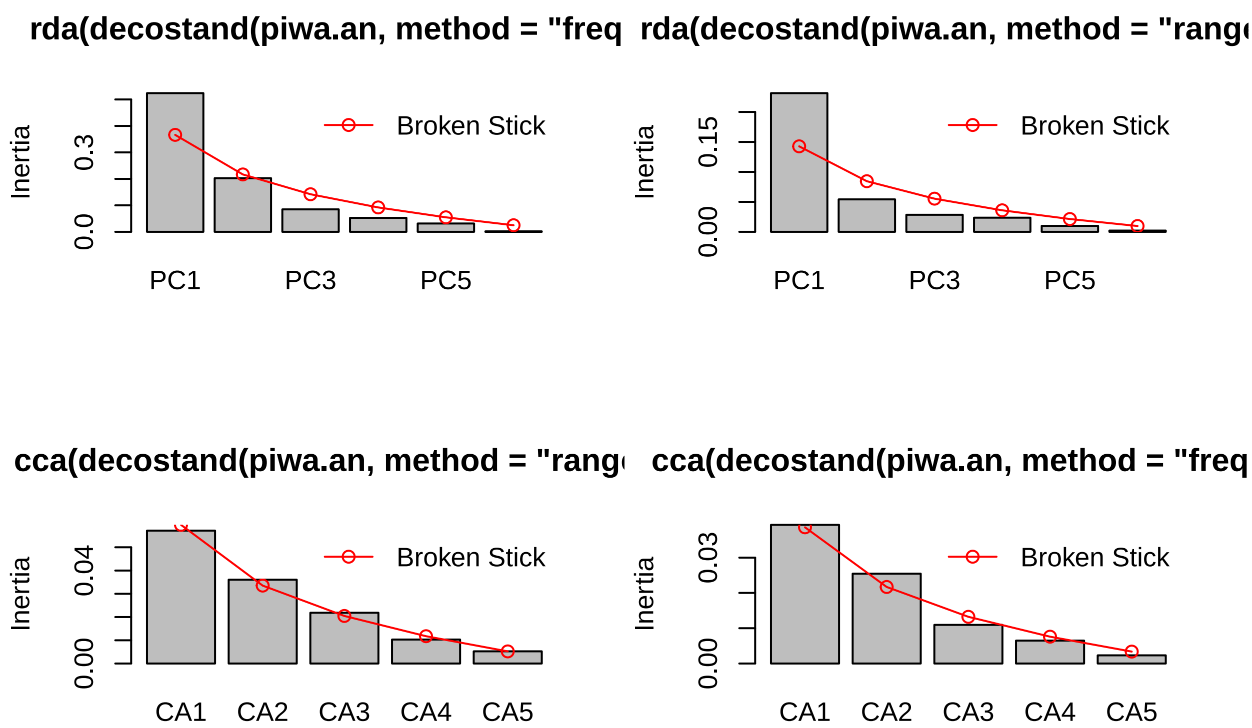
```
## Call: cca(X = decostand(piwa.an, method = "freq"))
##
##               Inertia Rank
## Total         0.08448
## Unconstrained 0.08448    5
## Inertia is mean squared contingency coefficient
##
## Eigenvalues for unconstrained axes:
##     CA1     CA2     CA3     CA4     CA5
## 0.03928 0.02545 0.01094 0.00649 0.00232
```

Tutaj nie ma znów jasnej odpowiedzi - podobna ilość zmienności wyjaśniana przez obie osie. Obejrzyjmy wykresy:

```
par(mfrow=c(2,2))
plot(rda(decostand(piwa.an, method='freq')))
plot(rda(decostand(piwa.an, method='range')))
plot(cca(decostand(piwa.an, method='range')))
plot(cca(decostand(piwa.an, method='freq')))
```

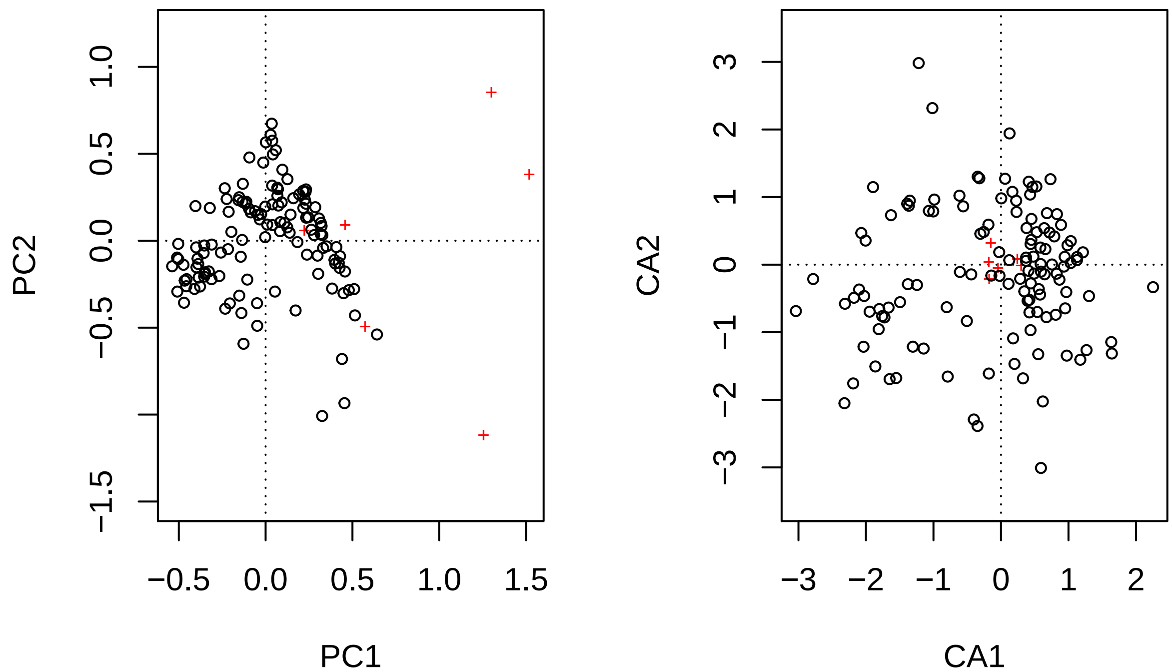


```
par(mfrow=c(2,2))
screeplot(rda(decostand(piwa.an, method='freq')),bstick=T)
screeplot(rda(decostand(piwa.an, method='range')),bstick=T)
screeplot(cca(decostand(piwa.an, method='range')),bstick=T)
screeplot(cca(decostand(piwa.an, method='freq')),bstick=T)
```



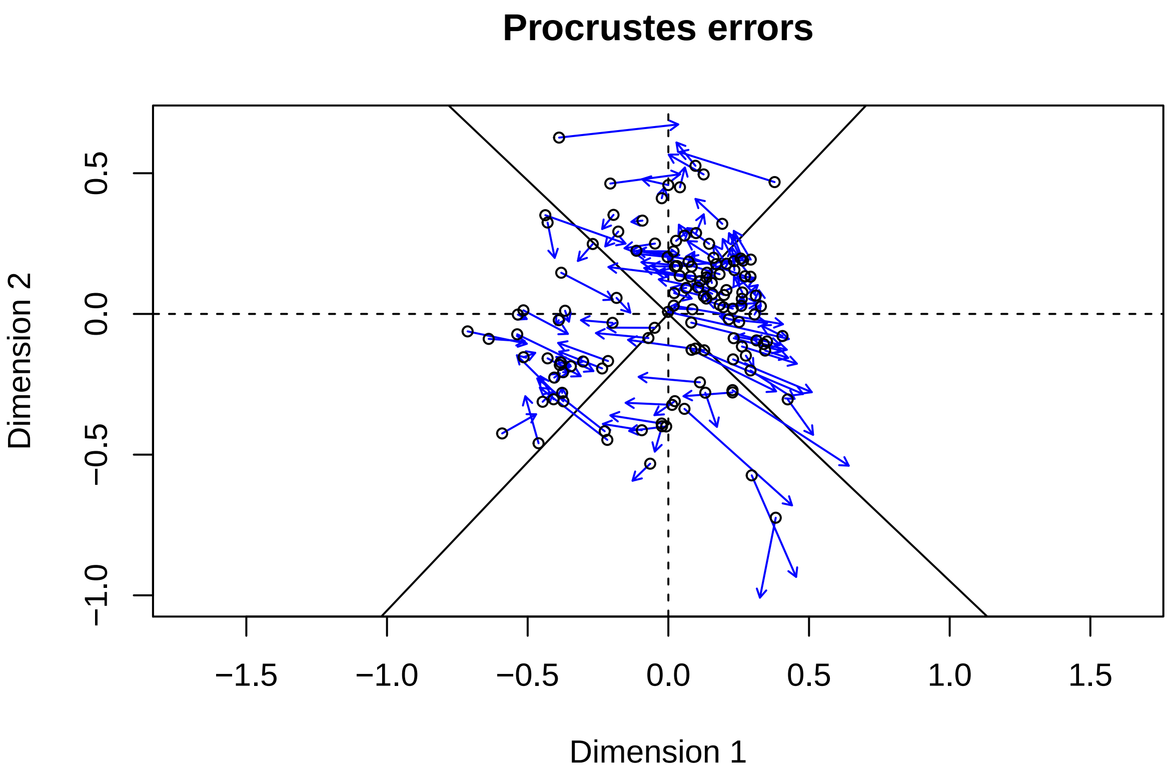
Analiza screeplotów wskazuje nam, że sensowne wyniki dadzą nam analizy po preprocessingu `freq` Przyjrzyjmy się tym wykresom znów bliżej:

```
par(mfrow=c(1,2))
plot(rda(decostand(piwa.an, method='freq')))
plot(cca(decostand(piwa.an, method='freq')))
```



Żeby sprawdzić czym się różnią od siebie analizy możemy wykonać wykres różnic w lokacji punktów za pomocą funkcji `procrusters()`:

```
par(mfrow=c(1,1))
plot(procrustes(rda(decostand(piwa.an, method='freq')),cca(decostand(piwa.an, method='freq'))))
```

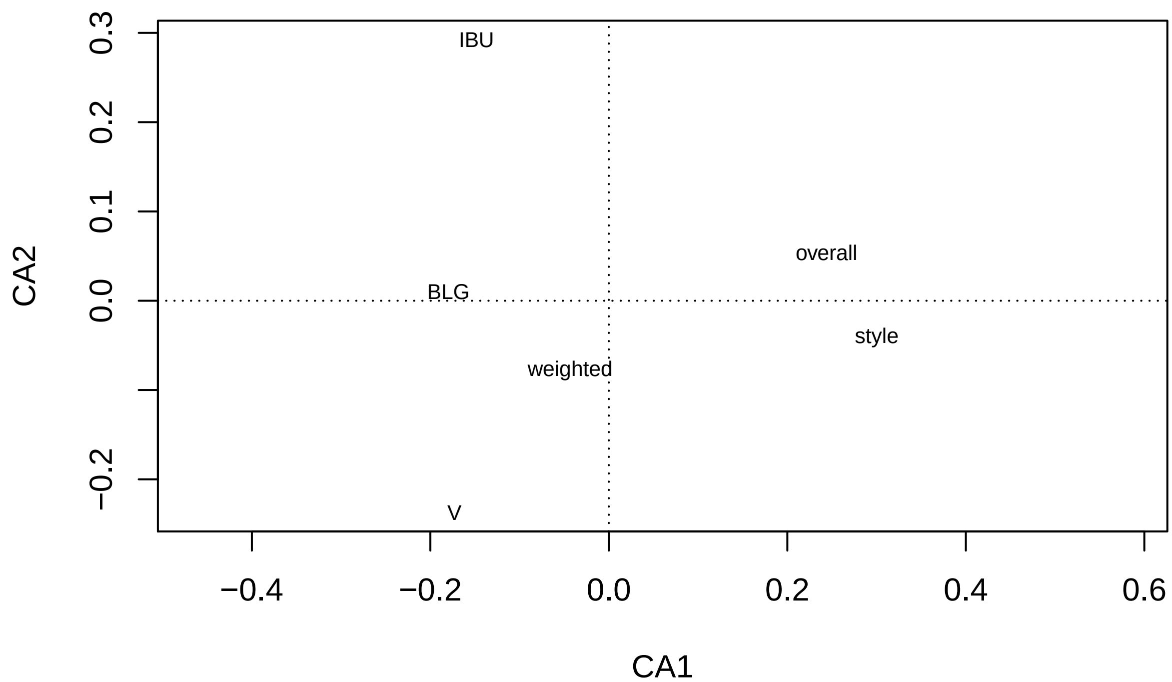


Funkcja ta pokazuje jak zmienia się lokacja tych samych punktów w dwóch różnych przestrzeniach ordynacyjnych. Tutaj widzimy duże zmiany. Do ostatecznej decyzji możemy wykorzystać ilość wyjaśnianej zmienności - w przypadku PCA oś PC1 wyjaśnia 0.05239/0.8969 = 58,4% a PC2 0.2027/0.8969=22,6% , natomiast w przypadku CA oś CA1 wyjaśnia 0.03928/0.08448=46,5% a oś CA2 0.02545/0.08448=30,1%. Różnice są na poziomie ok. 10%, więc można się zastanowić nad tym, który wynik jest interpretowalny. Wracamy do kształtu rozrzutu punktów - przy PCA mamy trójkąt, przy CA - mniej więcej równomierny rozrzut wzdłuż osi. Dodatkowo przy PCA cechy są w miejscach gdzie jest mało obiektów - to znaczy że coś tam nie do końca gra. Wybieramy coś, co ma większy sens - CA.

wizualizacja

Przyjrzyjmy się uzyskanym wynikom:

```
piwo.ca<-cca(decostand(piwa.an, method='freq'))
plot(piwo.ca, disp='species')
```

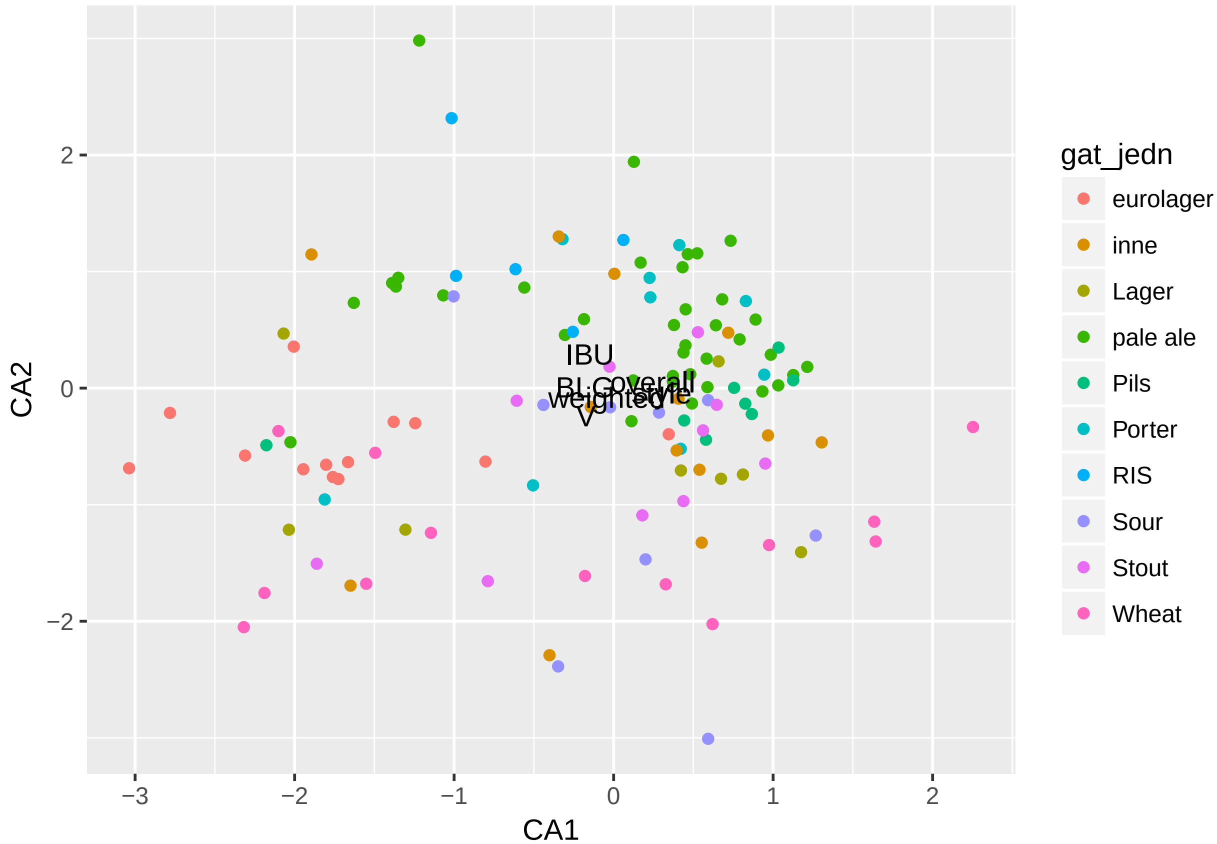


Na pierwszy rzut oka widzimy potwierdzenie różnic ocen stylu oraz całościowej. Widzimy też że IBU, V oraz BLG idą w jednej osi. Jest to dość logiczny wynik - aby uzyskać mocniejsze piwo potrzeba więcej ekstraktu. Aby piwo było zbalansowane - trzeba dodać do mocniejszego więcej chmielu. Nie widać jednak związku pomiędzy mocą piwa a jego oceną - wektory są prostopadłe do siebie. Pytanie jak wygląda zróżnicowanie stylów piwnych w przestrzeni ordynacyjnej. Stwórzmy znów dwa data.frame'y:

```
ggpiwo.obiekty<-as.data.frame(scores(piwo.ca,display='sites'))
ggpiwo.obiekty<-cbind(ggpiwo.obiekty, piwa)
ggpiwo.cechy<-as.data.frame(scores(piwo.ca,display='species'))
ggpiwo.cechy$name<-rownames(ggpiwo.cechy)
```

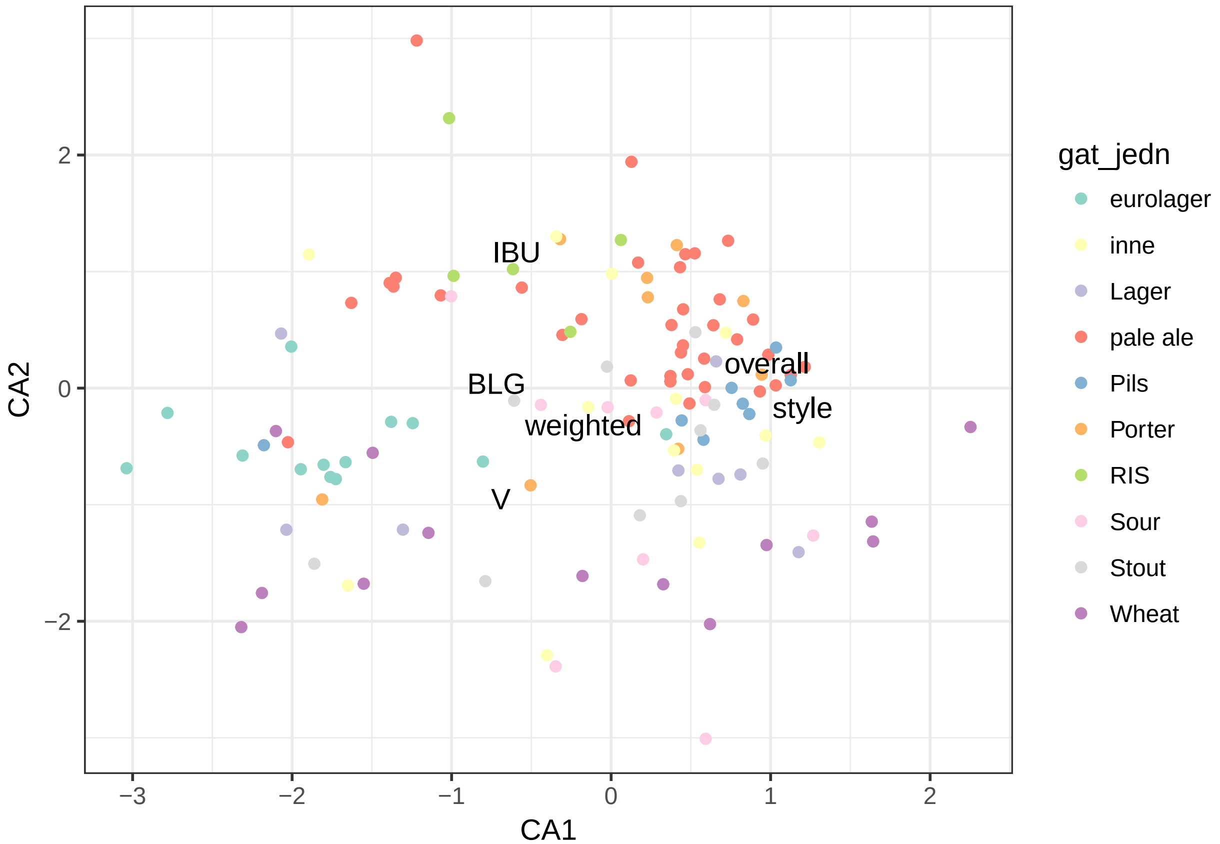
I zbudujmy wykres:

```
g<-ggplot(ggpiwo.obiekty, aes(x=CA1,y=CA2))
g<-g+geom_point(aes(col=gat_jedn))
g+geom_text(data=ggpiwo.cechy, aes(label=name))
```



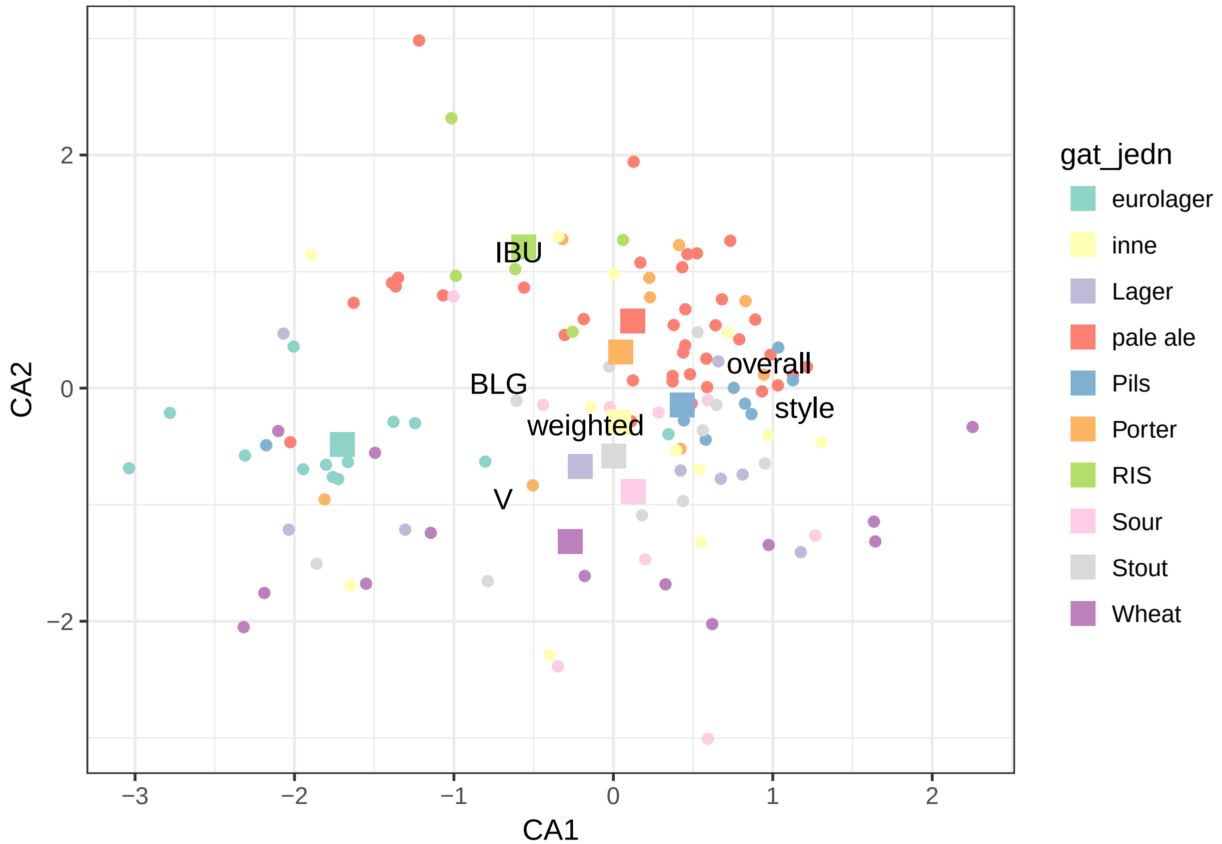
W celu lepszej wizualizacji możemy przeskalować sobie współrzędne cech - przemnożyć przez stałą, np. przez 4. Musimy jednak pamiętać że jest to poważna ingerencja w wynik analizy i dać o tym informację, np. w opisie ryciny. Z reguły PCA uwyplukla bardziej cechy a CA - obiekty, stąd przy CA dla lepszej wizualizacji różnic między cechami przemnożyliśmy to przez 4.

```
ggpiwo.cechy$CA1<-ggpiwo.cechy$CA1*4
ggpiwo.cechy$CA2<-ggpiwo.cechy$CA2*4
g<-ggplot(ggpiwo.obiekty, aes(x=CA1,y=CA2))
g<-g+geom_point(aes(col=gat_jedn))+geom_text(data=ggpiwo.cechy, aes(label=name))
g+scale_color_brewer(palette='Set3')+theme_bw()
```



Zachowaliśmy relacje między cechami i udało nam się ją zestawić na wykresie. Widzimy że niektóre style się wyróżniają. Możemy narysować centroidy stylów:

```
library(dplyr)
ggpiwo.obiekty.centro<-ggpiwo.obiekty%>%group_by(gat_jedn)%>%summarise(CA1=mean(CA1),
                CA2=mean(CA2))
p<-ggplot(ggpiwo.obiekty, aes(x=CA1,y=CA2))
p<-p+geom_point(aes(col=gat_jedn))
p<-p+geom_point(data=ggpiwo.obiekty.centro,aes(x=CA1,y=CA2,col=gat_jedn),size=4,shape=15)
p<-p+geom_text(data=ggpiwo.cechy, aes(label=name))
p<-p+scale_color_brewer(palette='Set3')+theme_bw()
p
```



Dzięki dodaniu symbolu centroidu widzimy w jaki sposób poszczególne obserwacje z grupy rozrzucone są wokół średniej. Bardzo duży rozrzut mamy w grupach 'inne' oraz 'pale ale'. Ogólne spojrzenie wskazuje, że mamy na dole piwa o małej goryczce - pszeniczne oraz kwasy - zwykle lepiej oceniane, oraz lagery i europlagery - zwykle oceniane gorzej. Portery oraz piwa typu pale ale (IPA, APA, AIPA) są z reguły oceniane lepiej, ale zdarzają się także słabsze IPY. Z reguły pilsy są oceniane lepiej niż lagery, co może wynikać z ich większego nasycenia chmielami i bardziej aromatycznej goryczki. Stosunkowo wysokie oceny dostaje też saison. Uzyskane wyniki wskazują jasno, że nie ma prostej recepty na doskonałe piwo, lecz że nawet w przypadku dobrze ocenianych styli zdarzają sie piwa słabsze.

6. Przypadek 3 - miasta

zbiór danych

```r
miasta<-read.csv('miasta.csv',sep=';')
summary(miasta)
```

```
##         nazwa          kraj          pop             perc_men     
##  Amsterdam : 1   Austria : 1   Min.   : 403166   Min.   :0.4400  
##  Ateny     : 1   Belgia  : 1   1st Qu.:1048193   1st Qu.:0.4675  
##  Berlin    : 1   Bułgaria: 1   Median :1578206   Median :0.4744  
##  Bratysława: 1   Czechy  : 1   Mean   :2120754   Mean   :0.4739  
##  Bruksela  : 1   Dania   : 1   3rd Qu.:2396340   3rd Qu.:0.4858  
##  Budapeszt : 1   Estonia : 1   Max.   :8100305   Max.   :0.4950  
##  (Other)   :17   (Other) :17                                      
##  urodzenia_rocznie Old.age.dependency.ratio       N        
##  Min.   :  4944    Min.   :14.80            Min.   :38.00  
##  1st Qu.: 13119    1st Qu.:21.55            1st Qu.:45.96  
##  Median : 18308    Median :23.30            Median :50.83  
##  Mean   : 23835    Mean   :24.50            Mean   :49.91  
##  3rd Qu.: 24068    3rd Qu.:27.50            3rd Qu.:54.01  
##  Max.   :139514    Max.   :36.30            Max.   :60.17  
##                                                            
##        E               elev            estab          agric_seminat   
##  Min.   :-9.183   Min.   :  0.0   Min.   :-1000.0   Min.   :0.1860  
##  1st Qu.: 4.625   1st Qu.: 24.5   1st Qu.:  654.0   1st Qu.:0.4333  
##  Median :16.367   Median : 70.0   Median : 1118.0   Median :0.4837  
##  Mean   :13.437   Mean   :125.5   Mean   :  836.6   Mean   :0.4880  
##  3rd Qu.:23.517   3rd Qu.:113.5   3rd Qu.: 1244.5   3rd Qu.:0.5657  
##  Max.   :26.100   Max.   :667.0   Max.   : 1550.0   Max.   :0.7586  
##                                                                    
##    lotniska            wbudowie          urban80_100      
##  Min.   :0.0005107   Min.   :0.0004021   Min.   :0.0009162  
##  1st Qu.:0.0017313   1st Qu.:0.0012058   1st Qu.:0.0047071  
##  Median :0.0032327   Median :0.0020656   Median :0.0102446  
##  Mean   :0.0042968   Mean   :0.0033641   Mean   :0.0224383  
##  3rd Qu.:0.0050846   3rd Qu.:0.0039595   3rd Qu.:0.0273681  
##  Max.   :0.0169758   Max.   :0.0119901   Max.   :0.1234866  
##                                                             
##   urban50_80       urban_10_30         urban30_50       
##  Min.   :0.00234   Min.   :0.0000187   Min.   :0.0002781  
##  1st Qu.:0.02166   1st Qu.:0.0042464   1st Qu.:0.0104544  
##  Median :0.03151   Median :0.0089812   Median :0.0173848  
##  Mean   :0.03607   Mean   :0.0152408   Mean   :0.0221040  
##  3rd Qu.:0.04418   3rd Qu.:0.0223722   3rd Qu.:0.0279006  
##  Max.   :0.10044   Max.   :0.0831962   Max.   :0.0713997  
##                                                           
##   urban_.10             drogi              lasy        
##  Min.   :0.0000000   Min.   :0.00884   Min.   :0.03921  
##  1st Qu.:0.0001571   1st Qu.:0.01748   1st Qu.:0.10134  
##  Median :0.0006517   Median :0.02533   Median :0.23006  
##  Mean   :0.0056978   Mean   :0.02813   Mean   :0.25988  
##  3rd Qu.:0.0038742   3rd Qu.:0.03411   3rd Qu.:0.40331  
##  Max.   :0.0429191   Max.   :0.05494   Max.   :0.56157  
```

```
##
##   tereny_ziel          przemysl       struktury_izolowane
##  Min.   :0.003641   Min.   :0.01250   Min.   :0.001037
##  1st Qu.:0.007685   1st Qu.:0.02774   1st Qu.:0.002275
##  Median :0.012227   Median :0.03567   Median :0.005436
##  Mean   :0.013322   Mean   :0.03830   Mean   :0.006142
##  3rd Qu.:0.016090   3rd Qu.:0.05031   3rd Qu.:0.008688
##  Max.   :0.033138   Max.   :0.08062   Max.   :0.017835
##
##    nieuzytki          wydobycia_skaldowanie     porty
##  Min.   :0.0003290   Min.   :0.001089      Min.   :0.000e+00
##  1st Qu.:0.0007246   1st Qu.:0.002360      1st Qu.:1.395e-05
##  Median :0.0011610   Median :0.003431      Median :5.491e-04
##  Mean   :0.0020201   Mean   :0.003699      Mean   :1.394e-03
##  3rd Qu.:0.0027896   3rd Qu.:0.005260      3rd Qu.:1.294e-03
##  Max.   :0.0085498   Max.   :0.007036      Max.   :1.072e-02
##
##  linie_kolejowe      sport_wypoczynek        wody
##  Min.   :0.0008352   Min.   :0.001391   Min.   :0.004068
##  1st Qu.:0.0016578   1st Qu.:0.005853   1st Qu.:0.009251
##  Median :0.0028510   Median :0.008299   Median :0.018319
##  Mean   :0.0029892   Mean   :0.010912   Mean   :0.035959
##  3rd Qu.:0.0041192   3rd Qu.:0.012713   3rd Qu.:0.032534
##  Max.   :0.0062353   Max.   :0.045362   Max.   :0.266840
##
##     pow_calk
##  Min.   : 107362
##  1st Qu.: 276187
##  Median : 424494
##  Mean   : 511342
##  3rd Qu.: 699238
##  Max.   :1745574
##
```

Zbiór danych `miasta` zawiera informacje o 23 europejskich stolicach. Głównym trzonem bazy są iformacje o strukturze użytkowania terenu wg Corine Land Cover urban atlas. Pozostałe zmienne mogą być pomocne w wyjaśnieniu zmienności strutury użytkowania terenu aglomeracji. Możemy spróbować wykonać ich projekcję pasywną lub zastosować metody ograniczone (bezpośrednie) ordynacji. Pierwszym krokiem będzie podział zbioru danych - wyciągniemy osobno proporcję poszczególnych form użytkowania terenu, a osobno - potencjalne czynniki objaśniające:

```r
miasta.clc<-miasta[,c(11:29)]
miasta.expl<-miasta[,c(3:10,30)]
rownames(miasta.clc)<-miasta$nazwa
rownames(miasta.expl)<-miasta$nazwa
```

wybór metody

Sprawdźmy jak długie są gradienty w miasta.clc. Może się wydawać że ponieważ nasze dane są wszystkie wyrażone procentowo to nie ma potrzeby transformacji. Jednakże zakres udziałów wskazuje, że niektóre czynniki będą miały zawyżone znaczenie. Przeskalujmy więc te czynniki w zakresie od 0 do 1 za pomocą `decostand()` i argumentu `method='range'`:

```
decorana(decostand(miasta.clc, method='range'))
```

```
##
## Call:
## decorana(veg = decostand(miasta.clc, method = "range"))
##
## Detrended correspondence analysis with 26 segments.
## Rescaling of axes with 4 iterations.
##
##                   DCA1    DCA2    DCA3    DCA4
## Eigenvalues     0.1864 0.09261 0.06794 0.04368
## Decorana values 0.1994 0.06602 0.03953 0.01516
## Axis lengths    1.8895 0.99922 1.23017 0.85449
```

Gradienty są krótkie, możemy więc zastosować większość analiz. Na początek spróbujmy analizę pośrednią - CA i PCA:

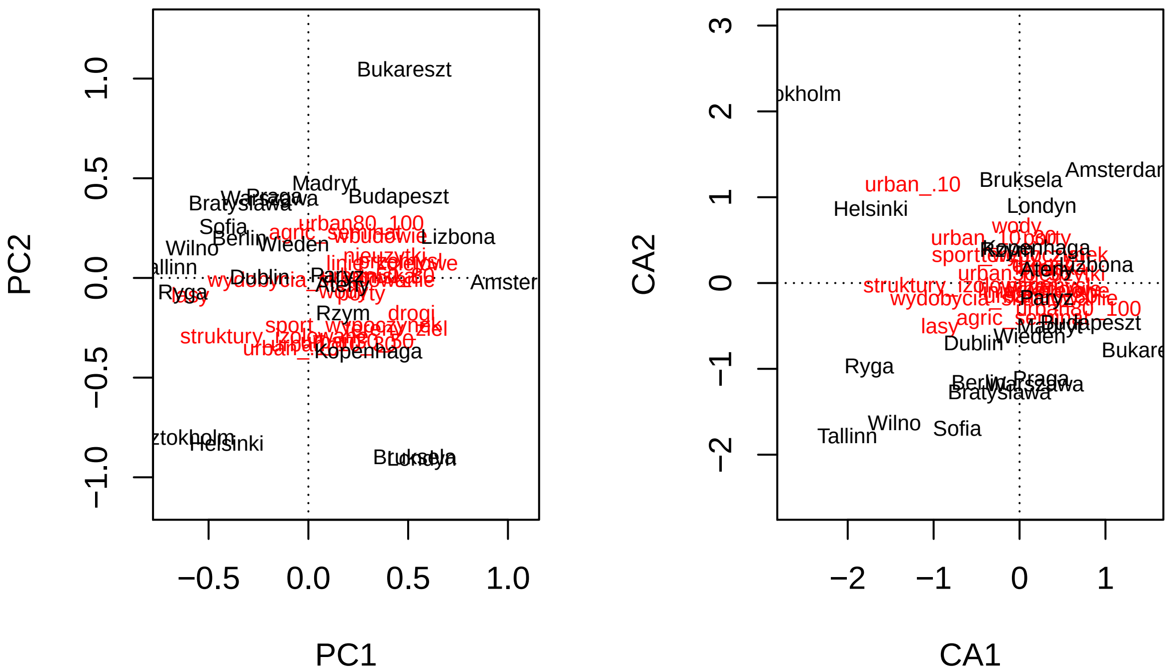
```
rda(decostand(miasta.clc, method = "range"))
```

```
## Call: rda(X = decostand(miasta.clc, method = "range"))
##
##               Inertia Rank
## Total           1.271
## Unconstrained   1.271   18
## Inertia is variance
##
## Eigenvalues for unconstrained axes:
##    PC1    PC2    PC3    PC4    PC5    PC6    PC7    PC8
## 0.5009 0.1935 0.1565 0.1353 0.0888 0.0533 0.0433 0.0346
## (Showed only 8 of all 18 unconstrained eigenvalues)
```

```
cca(decostand(miasta.clc, method = "range"))
```

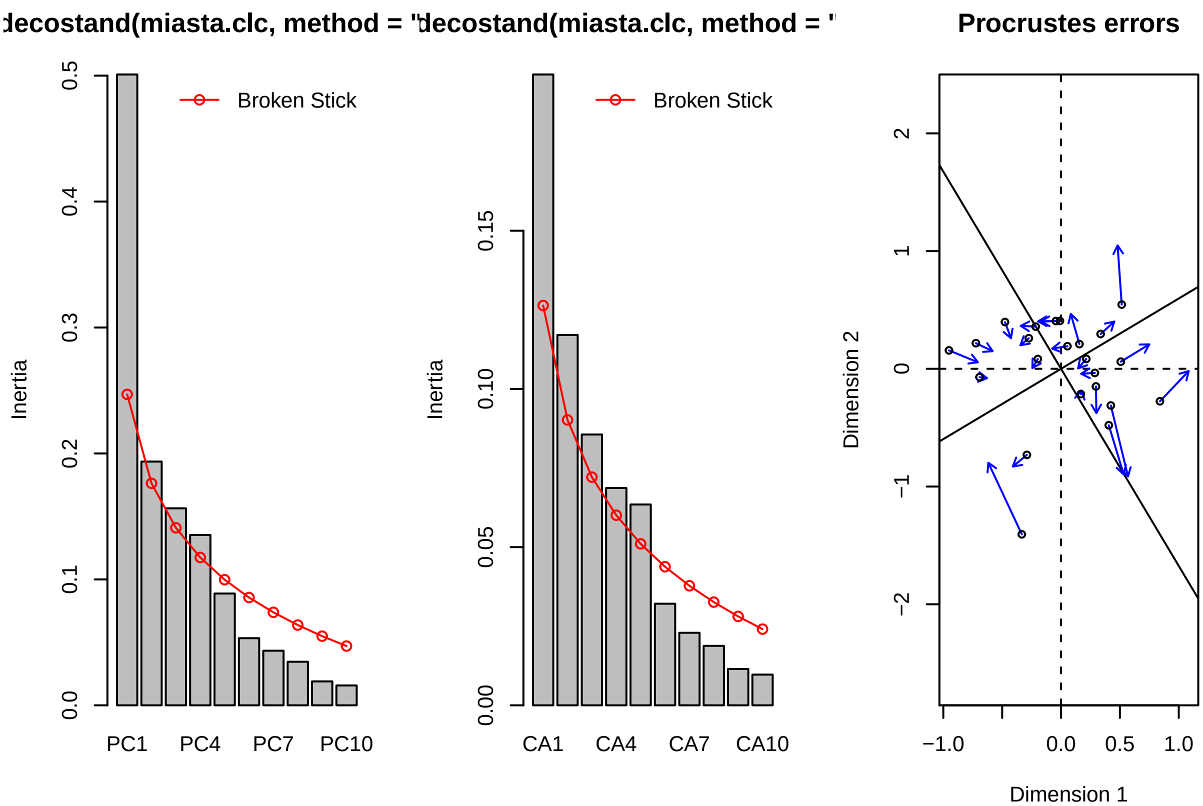
```
## Call: cca(X = decostand(miasta.clc, method = "range"))
##
##               Inertia Rank
## Total          0.6507
## Unconstrained  0.6507   18
## Inertia is mean squared contingency coefficient
##
## Eigenvalues for unconstrained axes:
##     CA1     CA2     CA3     CA4     CA5     CA6     CA7     CA8
## 0.19940 0.11706 0.08560 0.06869 0.06348 0.03210 0.02292 0.01879
## (Showed only 8 of all 18 unconstrained eigenvalues)
```

```
par(mfrow=c(1,2))
plot(rda(decostand(miasta.clc, method = "range")))
plot(cca(decostand(miasta.clc, method = "range")))
```


PC1
PC2
CA1
CA2
Bukareszt
Madryt
Budapeszt
Bratysława
Sofia
Berlin
Wilno
Wieden
Lizbona
Dublin
Paryz
Ateny
Amster
Ryga
Rzym
Kopenhaga
Helsinki
Bruksela
Londyn
urban80_100
nieuzytki
drogi
struktury_izolowane
lasy
Amsterdam
urban_.10
Londyn
Helsinki
Kopenhaga
Lizbona
Ryga
Praga
Warszawa
Tallinn
Wilno
Sofia
wody
sport
wydobycia
agric_
urban80_100

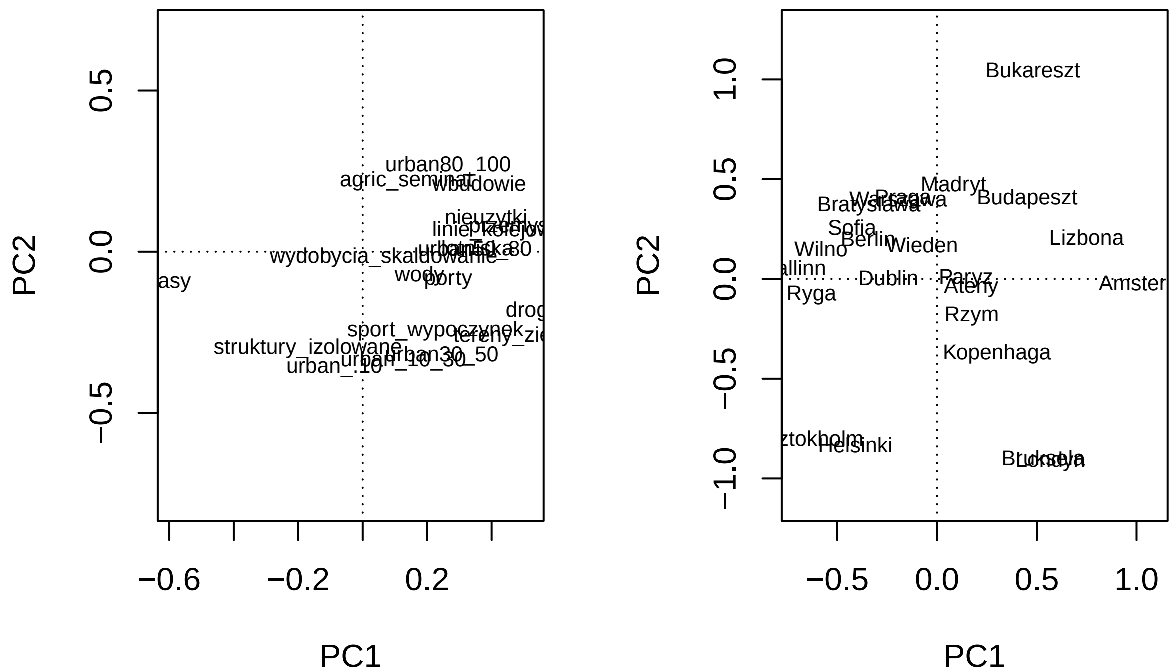
Sytuacja jest podobna do poprzedniej. Sprawdźmy znów screeploty i procrusters:

```
par(mfrow=c(1,3))
screeplot(rda(decostand(miasta.clc, method =
"range")),bstick=T)
screeplot(cca(decostand(miasta.clc, method =
"range")),bstick=T)
plot(procrustes(rda(decostand(miasta.clc, method =
"range")),cca(decostand(miasta.clc, method =
"range"))))
```



Różnice między analizami są dość spore. Screeploty dopuszczają użycie obu z nich. Proporcja zmienności wyjaśnianej przez pierwszą i drugą oś sugeruje użycie PCA.

```r
par(mfrow=c(1,2))
plot(rda(decostand(miasta.clc, method =
"range")),display='species')
plot(rda(decostand(miasta.clc, method =
"range")),display='sites')
```



Pierwszy rzut oka pozwala dostrzec główne źródła zmienności w badanym układzie: oś PC1 różnicuje nam miasta o dużym udziale lasów od tych, gdzie mamy dużo dróg, obszarów przemysłowych oraz terenów portowych. Oś PC2 różnicuje miasta o stosunkowo luźnej zabudowie (różne klasy gęstości zabudowy) od modelu miast z dużym zagęszczeniem zabudowy i obszarów półnaturalnych i rolniczych (agric_seminat).

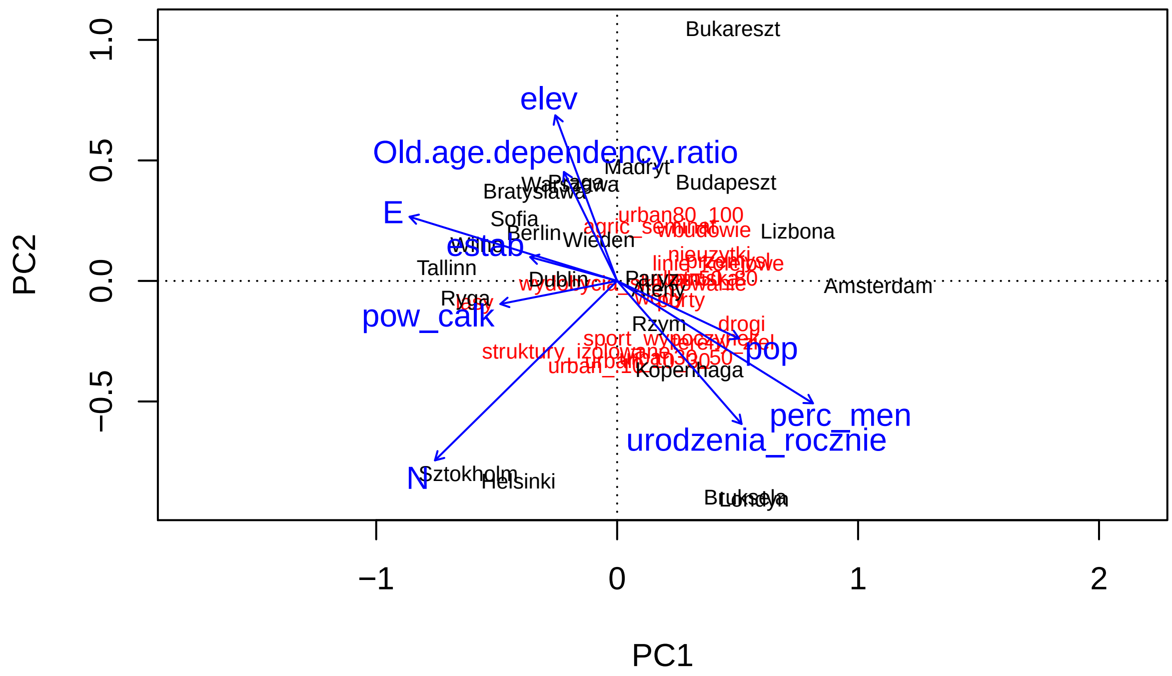
zmienne dodatkowe

W zasadzie moglibyśmy na tym poprzestać, przekształcając ten obrazek w zgrabnego ggplota. Pozostaje jednak głód wykorzystania reszty informacji. Mogłyby one podpowiedzieć nam dlaczego Sztokholm i Helsinki tak bardzo odbiegają od reszty. Albo czy uzyskany wynik ma jakieś powiązania geograficzne. Na początek spróbujmy pasywnej projekcji za pomocą funkcji `envfit()`:

```
pca.miasta<-rda(decostand(miasta.clc, method ="range"))
miasta.fit<-envfit(pca.miasta, miasta.expl)
miasta.fit
```

```
##
## ***VECTORS
##
##                                PC1      PC2     r2 Pr(>r)
## pop                        0.90419 -0.42714 0.1206  0.248
## perc_men                   0.84812 -0.52980 0.3537  0.017 *
## urodzenia_rocznie          0.65655 -0.75428 0.2384  0.067 .
## Old.age.dependency.ratio -0.43900  0.89848 0.0975  0.361
## N                         -0.71258 -0.70159 0.4335  0.004 **
## E                         -0.95546  0.29513 0.3132  0.021 *
## elev                      -0.34997  0.93676 0.2072  0.106
## estab                     -0.96427  0.26491 0.0540  0.576
## pow_calk                  -0.98103 -0.19387 0.0940  0.377
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Permutation: free
## Number of permutations: 999
```

```
plot(pca.miasta)
plot(miasta.fit)
```



Po dodaniu pasywnych wektorów możemy zauważyć kilka rzeczy. Po pierwsze, niewiele z nich przebiega w osiach PC1 i PC2. Ta informacja mówi nam, że mogą one wnieść coś nowego do układu jeśli zdecydujemy się na RDA. Po drugie - widzimy nowe zależności. Strzałki wskazują nam że w stronę prawego dolnego narożnika rośnie liczba mieszkańców. Z najludniejszych miast jest tam tylko Londyn, ale najmniej zaludnione miasta są po drugiej stronie strzałki. Stąd znacznie dłuższa strzałka i współczynnik determinacji udziału mężczyzn w populacji. Strzałka ta jest skierowana przeciwnie do strzałki E - długości geograficznej i odzwierciedla opisywany trend demograficzny związany z przewagą kobiet nad mężczyznami we wschodniej Europie. Widczony jest też związek pomiędzy rokiem założenia miasta `estab` a `E` i brak związku pomiędzy `estab` a `N` widoczny poprzez prostopadłe położenie strzałek. Z drugiej strony krótka strzałka oznacza słaby związek. Długa strzałka charakteryzuje wysokość n.p.m. `elev`, co jest związane z nadmorskim położeniem miejscowości znajdujących się naprzeciw strzałki.

Próbując dopasować w sposób aktywny wybrane wskaźniki do PCA aby uzyskać RDA musimy się zastanowić nad dwoma problemami: 1. Czy zmiana analizy na bardziej skomplikowaną poprawi jej jakość i wniesie coś nowego? 2. W jaki sposób wybrać poszczególne elementy modelu?

Pierwszy problem możemy rozwiązać poprzez porównanie AIC modelu zerowego (de facto PCA) do poszczególnych RDA. Aby określić AIC obiektu typu `rda` lub `cca` nie możemy wykorzystać bazowej funkcji `AIC()`, użyjemy po prostu funkcji `step()`.

```
rda0<-rda(decostand(miasta.clc,method='range')~1)
rda0
```

```
## Call: rda(formula = decostand(miasta.clc, method = "range") ~ 1)
##
##                Inertia Rank
## Total            1.271
## Unconstrained    1.271   18
## Inertia is variance
```

```
##
## Eigenvalues for unconstrained axes:
##    PC1    PC2    PC3    PC4    PC5    PC6    PC7    PC8
## 0.5009 0.1935 0.1565 0.1353 0.0888 0.0533 0.0433 0.0346
## (Showed only 8 of all 18 unconstrained eigenvalues)
```

```
#czyli jest to po prostu nasze PCA
step(rda0)
```

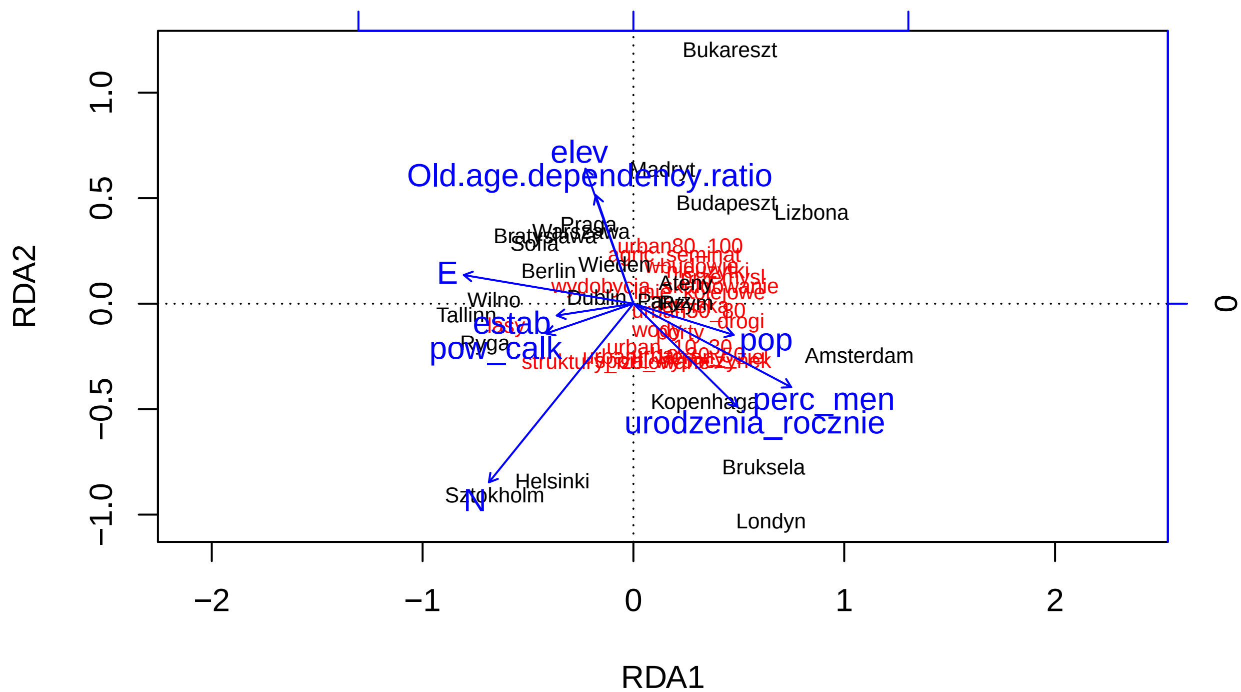
```
## Start:  AIC=6.5
## decostand(miasta.clc, method = "range") ~ 1

## Call: rda(formula = decostand(miasta.clc, method = "range") ~ 1)
##
##               Inertia Rank
## Total           1.271
## Unconstrained   1.271   18
## Inertia is variance
##
## Eigenvalues for unconstrained axes:
##    PC1    PC2    PC3    PC4    PC5    PC6    PC7    PC8
## 0.5009 0.1935 0.1565 0.1353 0.0888 0.0533 0.0433 0.0346
## (Showed only 8 of all 18 unconstrained eigenvalues)
```

na samej górze wyniku mamy `Start: AIC=6.5`. Mamy więc już punkt odniesienia. Drugi punkt jest bardziej skomplikowany: trzeba wybrać takie czynniki do modelu, aby nie były ze sobą skorelowane oraz żeby miały sens. Gdyby nie było tych ograniczeń moglibyśmy po prostu wrzucić wszystkie czynniki do modelu RDA i zastosować funkcję `step()` do wybrania modelu o najmniejszym AIC:

```
miasta.expl<-decostand(miasta.expl, method = "range")
rda.all<-rda(decostand(miasta.clc, method = "range") ~., data = miasta.expl)
rda.all
```

```
## Call: rda(formula = decostand(miasta.clc, method = "range") ~ pop
## + perc_men + urodzenia_rocznie + Old.age.dependency.ratio + N + E
## + elev + estab + pow_calk, data = miasta.expl)
##
##               Inertia Proportion Rank
## Total          1.2715     1.0000
## Constrained    0.7780     0.6119    9
## Unconstrained  0.4935     0.3881   13
## Inertia is variance
##
## Eigenvalues for constrained axes:
##   RDA1   RDA2   RDA3   RDA4   RDA5   RDA6   RDA7   RDA8   RDA9
## 0.4188 0.1577 0.0875 0.0501 0.0255 0.0186 0.0111 0.0075 0.0011
##
## Eigenvalues for unconstrained axes:
##     PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9
## 0.15671 0.10965 0.06351 0.05791 0.03348 0.03098 0.01657 0.01054 0.00518
##    PC10    PC11    PC12    PC13
## 0.00368 0.00328 0.00135 0.00065
```

```r
plot(rda.all)
```




```r
rda.stepmodel<-step(rda.all)
```

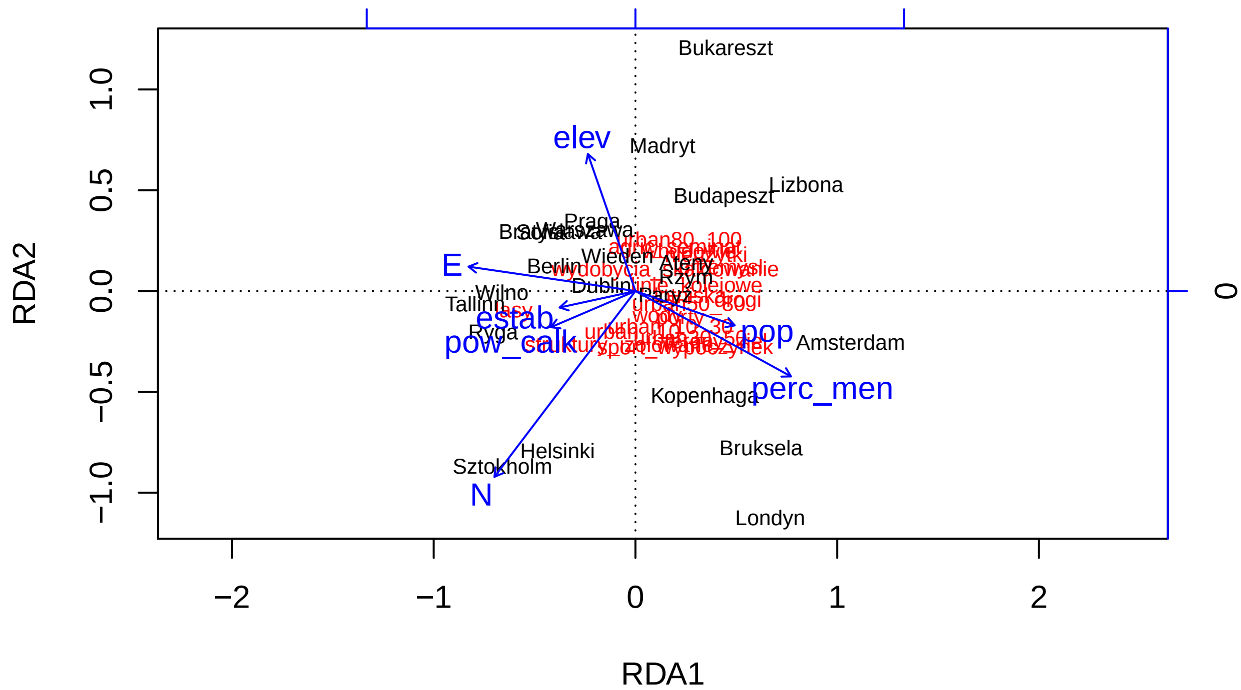
```
## Start:  AIC=2.73
## decostand(miasta.clc, method = "range") ~ pop + perc_men + urodzenia_rocznie +
##     Old.age.dependency.ratio + N + E + elev + estab + pow_calk
##
##                            Df    AIC
## - urodzenia_rocznie         1 1.7672
## - Old.age.dependency.ratio  1 2.4285
## <none>                        2.7348
## - E                         1 2.8579
## - pop                       1 2.8612
## - perc_men                  1 3.3387
## - estab                     1 3.7748
## - elev                      1 3.8643
## - pow_calk                  1 5.2275
## - N                         1 5.5938
##
## Step:  AIC=1.77
## decostand(miasta.clc, method = "range") ~ pop + perc_men + Old.age.dependency.ratio +
##     N + E + elev + estab + pow_calk
##
##                            Df    AIC
## - Old.age.dependency.ratio  1 1.3339
```

```
## <none>                        1.7672
## - E                         1 1.7861
## - perc_men                  1 2.1280
## - pop                       1 2.2923
## - elev                      1 2.7344
## - estab                     1 3.0895
## - pow_calk                  1 3.8629
## - N                         1 5.0266
##
## Step:  AIC=1.33
## decostand(miasta.clc, method = "range") ~ pop + perc_men + N +
##     E + elev + estab + pow_calk
##
##            Df    AIC
## <none>        1.3339
## - E         1 1.4858
## - pop       1 1.8649
## - perc_men  1 1.9154
## - elev      1 2.0756
## - estab     1 2.5276
## - pow_calk  1 3.5107
## - N         1 4.2065
```

```r
rda.stepmodel
```

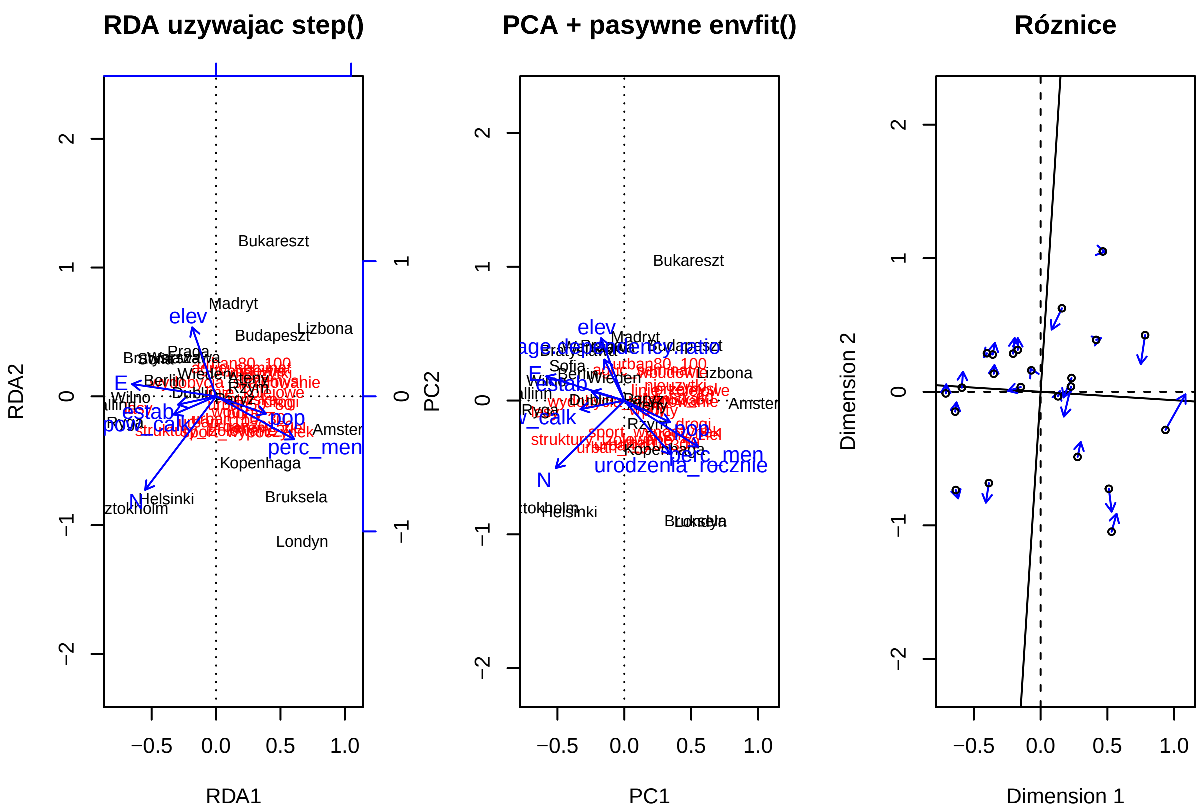
```
## Call: rda(formula = decostand(miasta.clc, method = "range") ~ pop
## + perc_men + N + E + elev + estab + pow_calk, data = miasta.expl)
##
##               Inertia Proportion Rank
## Total          1.2715     1.0000
## Constrained    0.7189     0.5654    7
## Unconstrained  0.5526     0.4346   15
## Inertia is variance
##
## Eigenvalues for constrained axes:
##   RDA1   RDA2   RDA3   RDA4   RDA5   RDA6   RDA7
## 0.4163 0.1463 0.0811 0.0363 0.0217 0.0111 0.0061
##
## Eigenvalues for unconstrained axes:
##     PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9
## 0.16395 0.11336 0.09299 0.06121 0.03762 0.03379 0.01757 0.01140 0.00821
##    PC10    PC11    PC12    PC13    PC14    PC15
## 0.00429 0.00344 0.00254 0.00137 0.00065 0.00014
```

```r
plot(rda.stepmodel)
```



Mimo redukcji AIC do 1.33 uzyskany wynik niewiele różni się od PCA z nałożonymi w sposób pasywny wektorami.

```
par(mfrow=c(1,3))
plot(rda.stepmodel, main='RDA używając step()')
plot(pca.miasta, main='PCA + pasywne envfit()')
plot(miasta.fit)
plot(procrustes(pca.miasta, rda.stepmodel),main='Różnice')
```



Potencjalnym zagrożeniem jest inflacja wariancji związana z wzajemnym skorelowaniem poszczególnych predyktorów. Możemy sprawdzić współczynniki inflacji wariancji - tzw. VIF za pomocą funkcji `vif.cca()`:

```
vif.cca(rda.all)
```

```
##                     pop                 perc_men        urodzenia_rocznie
##                7.004990                 2.617935                 4.363063
## Old.age.dependency.ratio                        N                        E
##                2.469366                 5.008702                 1.904477
##                    elev                    estab                 pow_calk
##                1.635596                 2.279127                 3.224568
```

```
vif.cca(rda.stepmodel)
```

```
##      pop perc_men        N        E     elev    estab pow_calk
## 2.427571 1.440321 3.141778 1.792255 1.575450 2.187920 2.030561
```

Nie przekraczają one 10, co jest traktowane jako reguła kciuka. Z drugiej strony inna reguła kciuka mówi że im więcej dodanych do formuły czynników, tym bardziej analiza ograniczona (constrained) zbliża się do niegraniczonej (unconstrained). Tutaj w zasadzie mamy podobny wynik i jeśli nie chodzi nam o szczegółowe pytania badawcze, to widzimy że nie ma potrzeby wykonywania RDA. Pamiętajmy jednak że wykonaliśmy tę analizę w sposób bezmyślny - wrzucając wszystkie czynniki do modelu. Spróbujmy teraz wykonać tę samą analizę w sposób przemyślany. Patrząc na PCA z pasywnymi wektorami przyjmijmy hipotezę o wpływie niektórych czynników na strukturę form użytkowania terenu. Takimi czynnikami może być rok założenia miasta `estab`, wysokość n.p.m. `elev` czy liczba mieszkańców `pop`.

```
rda1<-rda(decostand(miasta.clc, method = "range") ~ pop + N + E + elev + estab, data = miasta.expl)
step(rda1)
```

```
## Start:  AIC=3.32
## decostand(miasta.clc, method = "range") ~ pop + N + E + elev +
##     estab
##
##         Df    AIC
## - pop    1 2.3089
## <none>     3.3157
## - estab  1 3.7004
## - E      1 4.0602
## - elev   1 4.7565
## - N      1 6.6395
##
## Step:  AIC=2.31
## decostand(miasta.clc, method = "range") ~ N + E + elev + estab
##
##         Df    AIC
## <none>     2.3089
## - estab  1 2.8167
## - elev   1 3.6155
## - E      1 3.7634
## - N      1 5.4576
```

```
## Call: rda(formula = decostand(miasta.clc, method = "range") ~ N +
## E + elev + estab, data = miasta.expl)
##
##               Inertia Proportion Rank
## Total          1.2715     1.0000
## Constrained    0.5232     0.4115    4
## Unconstrained  0.7483     0.5885   18
## Inertia is variance
##
## Eigenvalues for constrained axes:
##    RDA1    RDA2    RDA3    RDA4
## 0.31606 0.12073 0.06508 0.02129
##
## Eigenvalues for unconstrained axes:
##     PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8
## 0.25843 0.13332 0.11431 0.07021 0.05458 0.03779 0.02395 0.01875
## (Showed only 8 of all 18 unconstrained eigenvalues)
```

Widzimy, że AIC= 3.32. Funkcja step podpowiada nam, że usunięcie zmiennej `pop` zmniejszy AIC do 2.31. Zróbmy więc taki model:

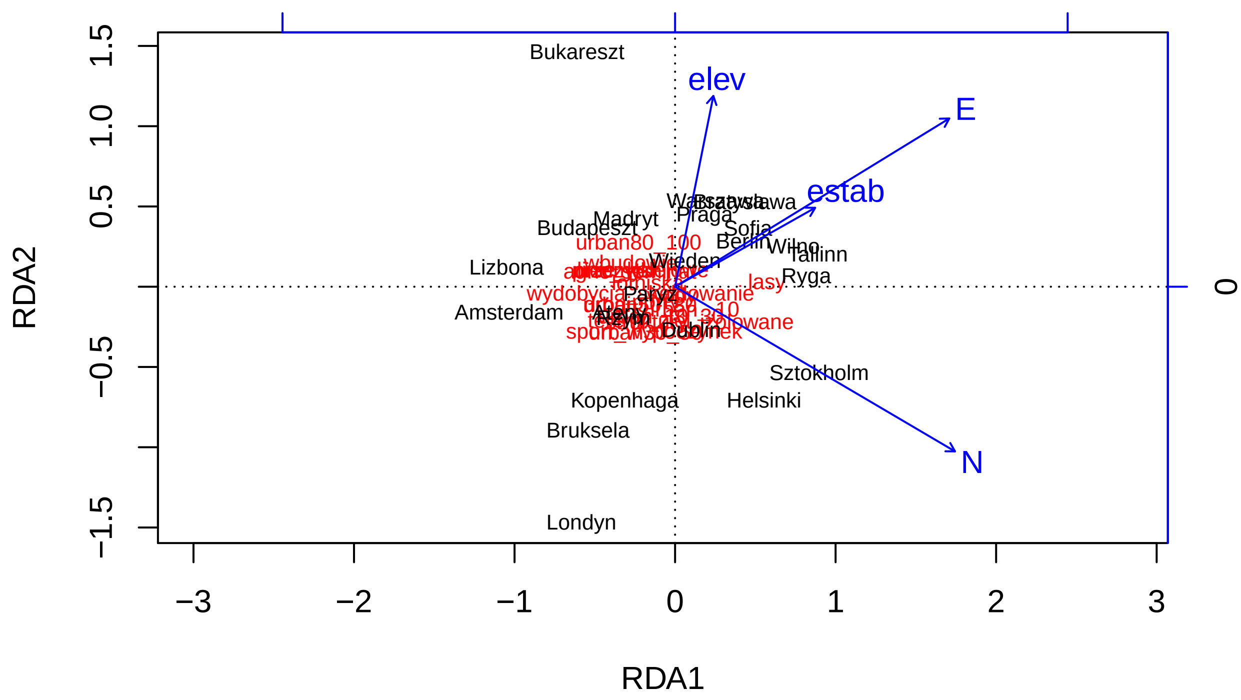
```r
rda2<-rda(decostand(miasta.clc, method = "range") ~ + N + E + elev + estab, data = miasta.expl)
step(rda2)
```

```
## Start:  AIC=2.31
## decostand(miasta.clc, method = "range") ~ +N + E + elev + estab
##
##         Df    AIC
## <none>     2.3089
## - estab  1 2.8167
## - elev   1 3.6155
## - E      1 3.7634
## - N      1 5.4576
```

```
## Call: rda(formula = decostand(miasta.clc, method = "range") ~ +N +
## E + elev + estab, data = miasta.expl)
##
##               Inertia Proportion Rank
## Total          1.2715     1.0000
## Constrained    0.5232     0.4115    4
## Unconstrained  0.7483     0.5885   18
## Inertia is variance
##
## Eigenvalues for constrained axes:
##    RDA1    RDA2    RDA3    RDA4
## 0.31606 0.12073 0.06508 0.02129
##
## Eigenvalues for unconstrained axes:
##     PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8
## 0.25843 0.13332 0.11431 0.07021 0.05458 0.03779 0.02395 0.01875
## (Showed only 8 of all 18 unconstrained eigenvalues)
```

tak przygotowany model dzieli zmienność uzyskaną dzięki dodanym czynnikom i zmienność z PCA. Dodane czynniki wyjaśniają mniej zmiennosci niż reszta i jest to zadowalająca propocja, choć mogła by być niższa. Zobaczmy jak wygląda wynik naszej analizy:

```r
plot(rda2)
```



Dodanie współrzędnych geograficznych spowodowało że miasta rozłożyły nam się przestrzenie w sposób geograficzny. Z drugiej strony zmniejszyło znaczenie klas pokrycia terenu. W związku z tym ich dodanie budzi nasze wątpliwości. Po usunięciu z modelu otrzymamy. . .

```r
rda3<-rda(decostand(miasta.clc, method = "range") ~ elev + estab, data = miasta.expl)
step(rda3)
```

```
## Start:  AIC=7.58
## decostand(miasta.clc, method = "range") ~ elev + estab
##
##         Df    AIC
## - estab  1 7.0728
## - elev   1 7.1107
## <none>     7.5773
##
## Step:  AIC=7.07
## decostand(miasta.clc, method = "range") ~ elev
##
##        Df    AIC
## - elev  1 6.5017
## <none>    7.0728
##
## Step:  AIC=6.5
## decostand(miasta.clc, method = "range") ~ 1
```

```
## Call: rda(formula = decostand(miasta.clc, method = "range") ~ 1,
## data = miasta.expl)
##
##               Inertia Rank
## Total           1.271
## Unconstrained   1.271   18
## Inertia is variance
##
## Eigenvalues for unconstrained axes:
##    PC1    PC2    PC3    PC4    PC5    PC6    PC7    PC8
## 0.5009 0.1935 0.1565 0.1353 0.0888 0.0533 0.0433 0.0346
## (Showed only 8 of all 18 unconstrained eigenvalues)
```

… informację, że lepszy jest model zerowy. O efekt współrzędnych geograficznych nie zawsze nam chodzi, więc niekoniecznie będziemy chcieli żeby to one decydowały o wyniku. W związku z tym wracamy do PCA z dopasowanymi pasywnie wektorami

wizualizacja

Wydobądźmy współrzędne dla cech i obiektów:

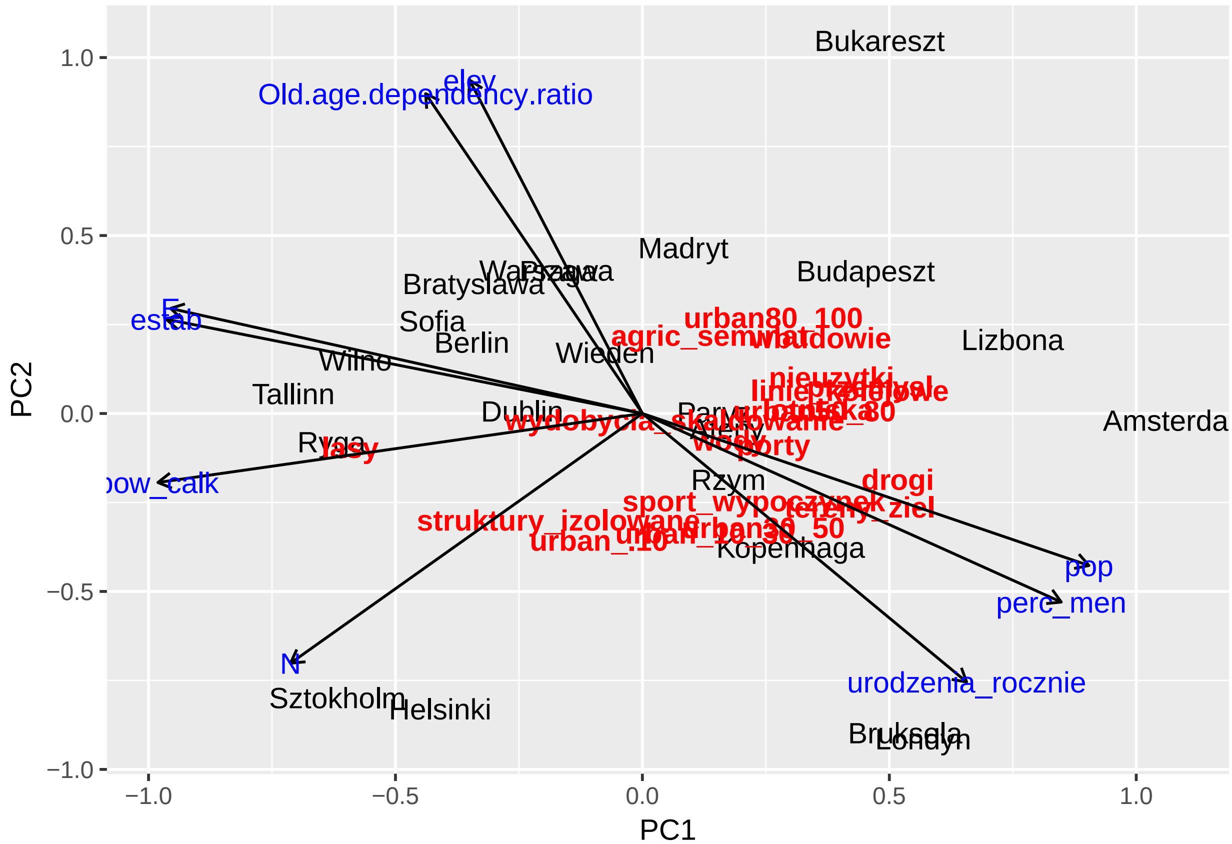
```
ggmiasto.obiekty<-as.data.frame(scores(pca.miasta,display='sites'))
ggmiasto.obiekty<-cbind(ggmiasto.obiekty, miasta.expl)
ggmiasto.obiekty$nazwa<-miasta$nazwa
ggmiasto.cechy<-as.data.frame(scores(pca.miasta,display='species'))
ggmiasto.cechy$name<-rownames(ggmiasto.cechy)
#teraz ggplot
p<-ggplot(ggmiasto.obiekty, aes(x=PC1, y=PC2))+geom_text(aes(label=nazwa))
p<-p+geom_text(data=ggmiasto.cechy, aes(label=name),col='red',fontface='bold')
```

Czas wyekstrachować strzałki. Tutaj z uwagi na bardziej złożoną konstrukcję obiektu typu `vectorfit` musimy się bardziej nagimnastykować:

```
ggmiastafit<-data.frame(nazwa=rownames(miasta.fit$vectors$arrows),
PC1=miasta.fit$vectors$arrows[,1],
PC2=miasta.fit$vectors$arrows[,2])
#na tym można by skończyć, ale niektórzy fetyszyzują p-values i r2:
ggmiastafit$r2<-miasta.fit$vectors$r
ggmiastafit$pval<-miasta.fit$vectors$pvals
```

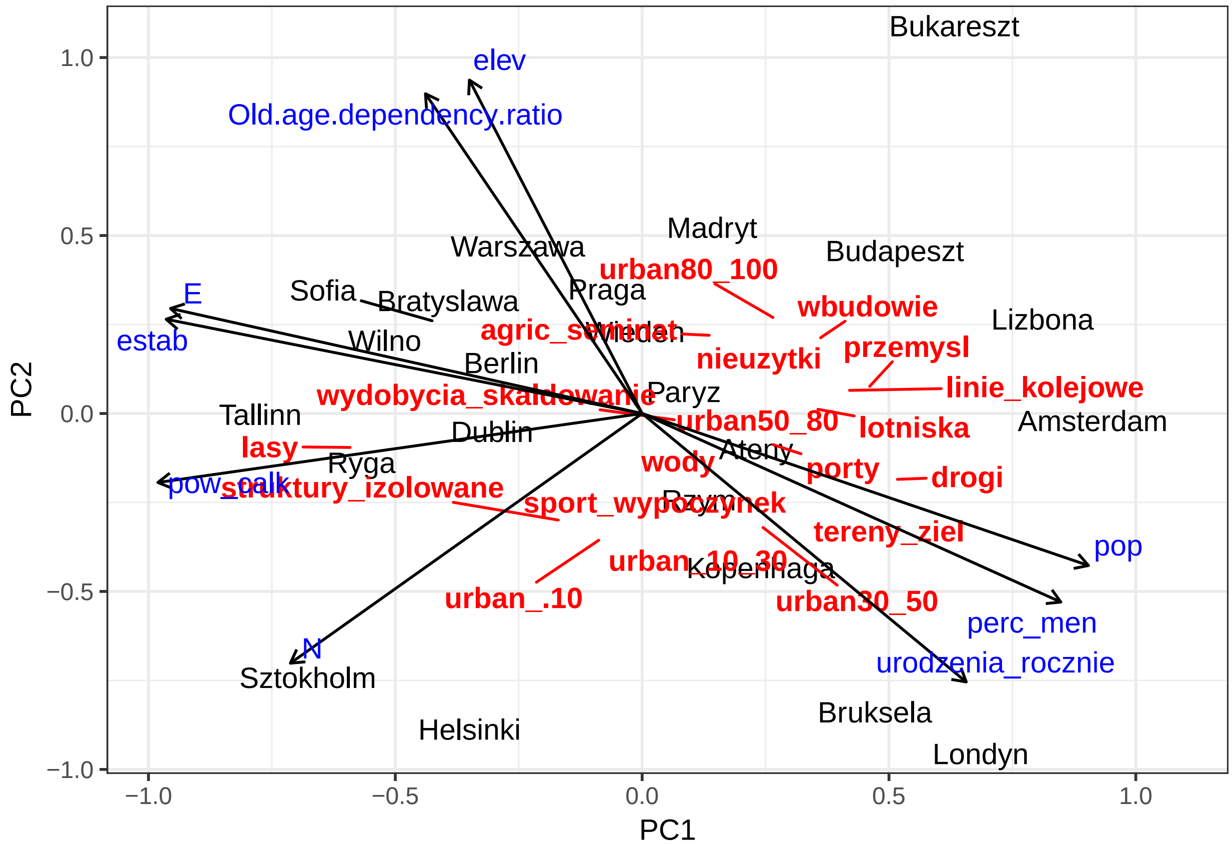
Istnieje rekomendacja pokazywania na wykresach tylko tych strzałek, których dopasowanie do wyniku analizy jest istotne statystycznie na podstawie p<0,05. P i R2 są określane na podstawie testów permutacyjnych. Przy 23 obserwacjach można przyjąć wyższy limit dla p, ale to zależy od specyfiki dziedziny.

```
p<-ggplot(ggmiasto.obiekty, aes(x=PC1, y=PC2))
p<-p+geom_text(aes(label=nazwa))
p<-p+geom_text(data=ggmiasto.cechy, aes(label=name),col='red',fontface='bold')
p<-p+geom_segment(data=ggmiastafit, aes(x=0,xend=PC1,y=0,yend=PC2),
arrow = arrow(length = unit(0.2,"cm")))
p+geom_text(data=ggmiastafit,aes(label=nazwa),col='blue')
```



Mamy więc wynik, widzimy jakie są relacje między zmiennymi i jakie są podobnieństwa między obiektami. Brakuje nam tylko trochę estetyki. Skorzystajmy z dobrodziejstwa biblioteki `ggrepel`:

```
library(ggrepel)
p<-ggplot(ggmiasto.obiekty, aes(x=PC1, y=PC2))
p<-p+geom_text_repel(aes(label=nazwa))
p<-p+geom_text_repel(data=ggmiasto.cechy, aes(label=name),col='red',fontface='bold')
p<-p+geom_segment(data=ggmiastafit, aes(x=0,xend=PC1,y=0,yend=PC2),arrow =
arrow(length = unit(0.2,"cm")))
p+geom_text_repel(data=ggmiastafit,aes(label=nazwa),col='blue')+theme_bw()
```



Zastapienie funkcji `geom_text()` funkcją `geom_text_repel()` powoduje rozsunięcie etykiet w celu zwiększenia czytelności. W przypadku gdy etykieta jest zbyt bardzo odsunięta od punktu to dorysowywana jest kreska. W ten sposób łatwo uporządkować obrazki z dużą liczbą napisów.

7. Podsumowanie + dalsze informacje

Metody ordynacji mogą pomóc szukać zależności pomiędzy zmiennymi w zbiorach danych. Same w sobie mogą stać się narzędziem wnioskowania lub mogą pokazać możliwości dalszej obróbki np. za pomocą modeli pozwalających na predykcję. Niezależnie od tego, czy przeprowadzone analizy będą głównym wynikiem naszej pracy, czy tylko krokiem na drodze do bardziej wyszukanych metod. Kluczem do najoptymalniejszego wyboru metod ordynacyjnych i ich interpretacji jest rozumienie swojego zbioru danych. Analizowane podczas warsztatów zbiory danych dla każdego uczestnika miały różny poziom trudności. Dlatego też każdy widział inne potencjalne manamenty i zalety dla poszczególnych prób analizy. Mimo opracowania różnych narzędzi pomocniczych i dobrych praktyk w ordynacji nadal wiele decyzji podejmujemy "na wyczucie". Z tego względu znajomość i rozumienie danych oraz oglądanie obrazków są podstawowym narzędziem wyboru analiz. Przytaczane w opracowaniu "reguły kciuka" nie zawsze dają najlepszy wynik. Wynika to z faktu że analizujemy zwykle dane różnej jakości. Zgodnie z ogólnie uznawaną regułą GIGO - Garbage In, Garbage Out - im lepsze mamy dane tym większa szansa na znalezienie czegośc ciekawego. Przy interpretacji wykresów diagnostycznych i roboczych należy mieć na uwadze mądrość zawartą w jednym z cytatów z pakietu `fortunes`:

```
fortunes::fortune(105)
```

```
##
## A sufficiently trained statistician can read the vagaries of a Q-Q plot
## like a shaman can read a chicken's entrails, with a similar recourse to
## scientific principles. Interpreting Q-Q plots is more a visceral than an
## intellectual exercise. The uninitiated are often mystified by the process.
## Experience is the key here.
##    -- Department of Mathematics and Statistics, Murdoch University
##       StatsNotes
```

Gdzie szukać dalszej wiedzy? - garść linków

Tutorial do pakietu `vegan`

Przegląd metod wg Michaela W. Palmera

Opis metod wg Davida Zelenego

R labs for Community Ecologists

Biecek & Trajkowski: Na przełaj przed Data Mining