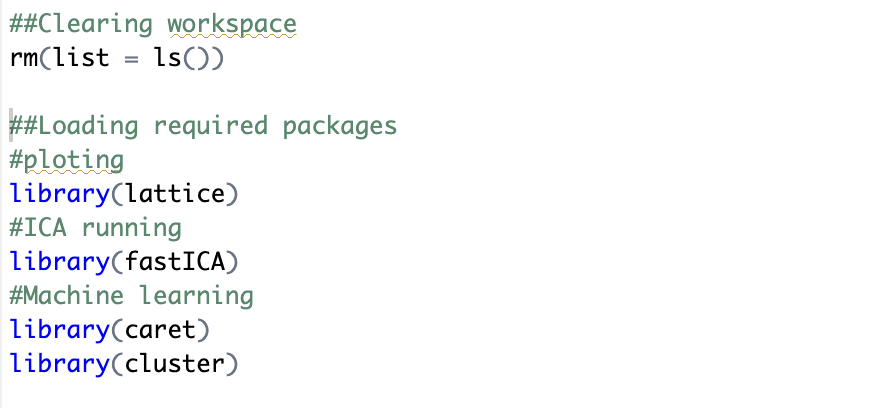
**جلسه ۱۰**

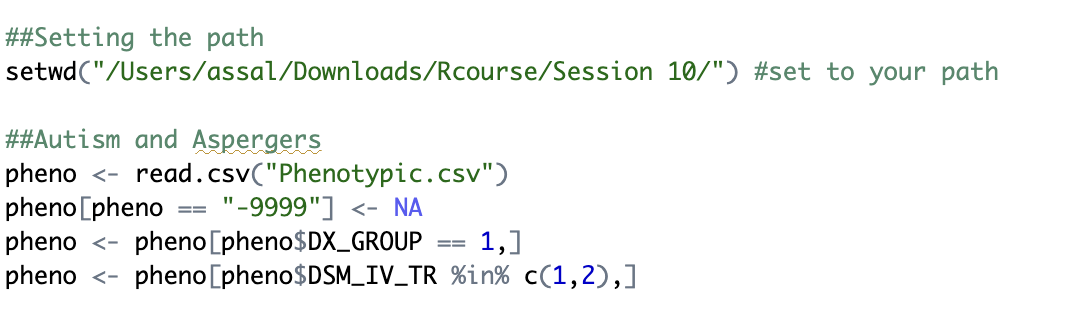
عسل سعیدی‌نیک، سپیده کاردان

در این جلسه موضوعات مختلفی را پوشش می‌دهیم و اولین مطلبی که به آن می‌پردازیم رگسیون غیر خطی می‌باشد.

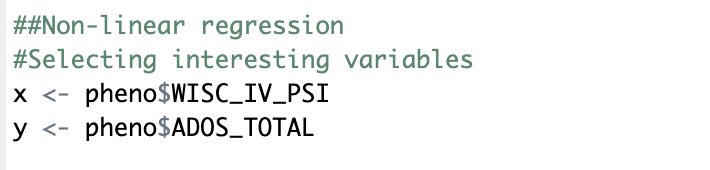
اول work space را پاک می‌کنیم و سپس کتابخانه‌های مورد نیاز این جلسه را نصب می‌کنیم:



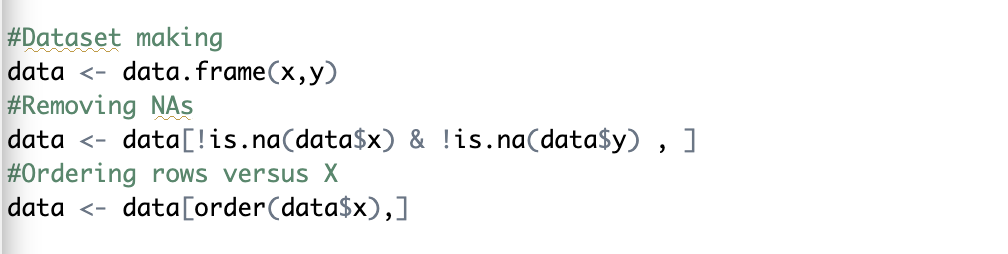
سپس working directory را روی مسیر مورد نظر set می‌کنیم و اوتیسم‌ها و آسپرگرها را از دیتا انتخاب می‌کنیم:



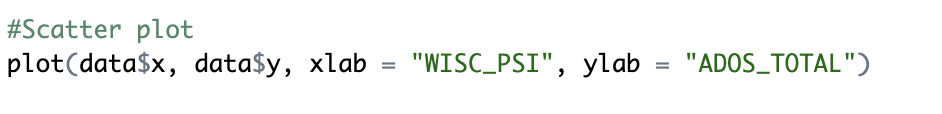
جلسه قبل ارتباط بین دو متغیر را با استفاده از رگرسیون خطی سنجیدیم. در این جلسه نیز می‌خواهیم ارتباط بین دو متغیر را بسنجیم، بنابراین باید دو متغیر x و y درست کنیم. یکی از این متغیرها ADOS\_TOTAL است که فاکتور کلی می‌باشد که به ما می‌گوید شدت اوتیسم چقدر است. متغیر دیگر تست وکسلر است که processing speed شرکت کننده‌ها را می‌سنجد. این دو متغیر را به این شکل درست می‌کنیم:

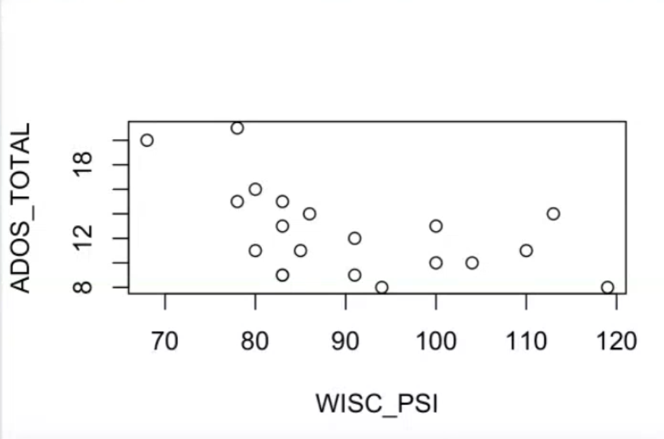


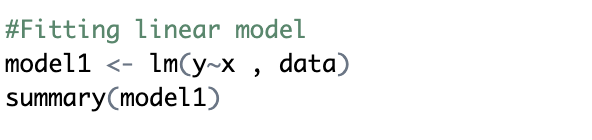
سپس یک data frame درست می‌کنیم، NAها را از مجموعه خارج می‌کنیم و سپس می‌خواهیم که داده‌ها بر اساس x مرتب شوند. یعنی متغیر x از کوچک به بزرگ مرتب شود و متغیر y متناظر با آن در رو به رویش قرار گیرد.



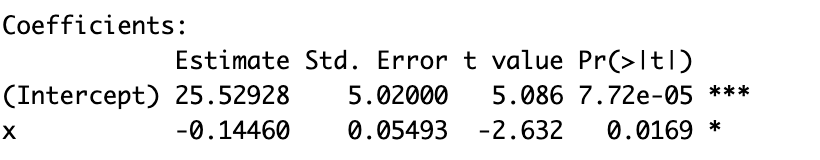
حالا یک scatter plot برای این متغیرها می‌کشیم. به نظر می‌رسد که رابطه بین دو متغیر خیلی خطی نمی‌باشد ولی اگر می‌خواست خطی باشد، ارتباطشان منفی می‌بود.



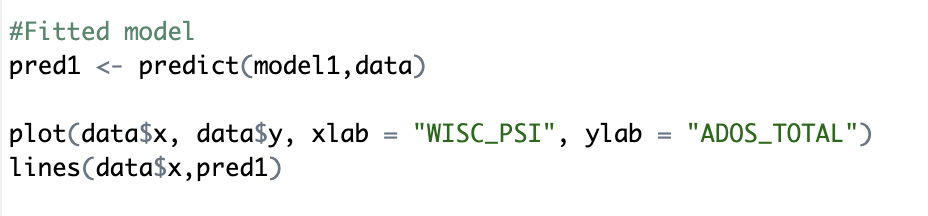
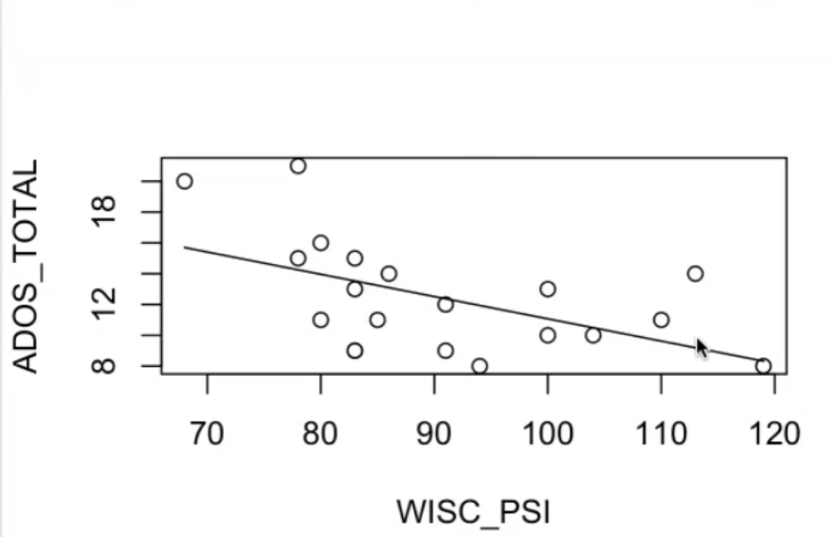


می‌توانیم یک خط به آن fit کنیم و ببینیم چه می‌شود:

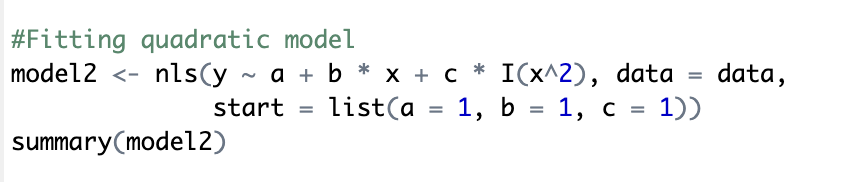
وقتی summary مدل را می‌گیریم می‌بینیم که خط معنادار نیز هست و می‌توان گفت یک رابطه خطی منفی وجود دارد.



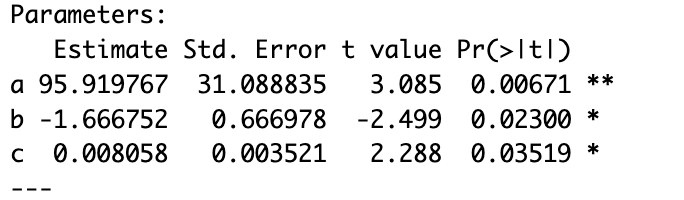
بر اساس مدلی که ساختیم یک پیش‌بینی درباره y انجام می‌دهیم. یعنی اینکه با استفاده از این مدل اگر x را بدهیم، yهای پیش‌بینی شده چه خواهند بود؟ سپس این خط جدید را plot می‌کنیم و أن را با استفاده از دستور lines روی scatterplot دیتا اصلی می‌اندازیم تا ببینیم مدل خطی که ساختیم چقدر به دیتا ما می‌آید. می‌بینیم که خیلی جور در نمی‌آید و انگار یک خط شکسته بیشتر به این data می‌خورد.



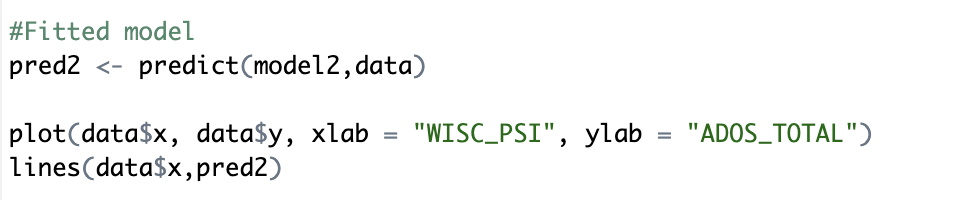
این دفعه سعی می‌کنیم یک تابع غیر خطی بکشیم و برای این کار باید از یک فرمول درجه دو استفاده کنیم که در آن x به توان ۲ می‌رسد و ۳ ثابت داریم (a و b و c) که همه آن‌ها را می‌نویسیم و از دستور nls استفاده می‌کنیم. البته b در واقع شیب خط است که در تابع درجه ۱ هم وجود دارد و ما فقط در فرمول آن را نمی‌نویسیم. همچنین موقع درست کردن این تابع باید بدانیم که باید به آن starting point بدهیم تا از آنجا شروع کند و بهترین تابع ممکن را به ما بدهد.

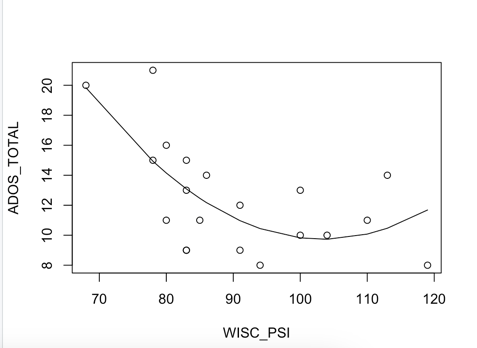


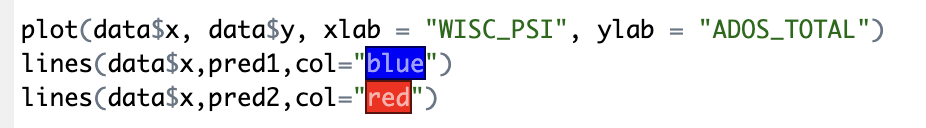
در این مدل‌ها وقتی summary می‌گیریم به ما p.value هر یک از ثابت‌ها را می‌دهد.

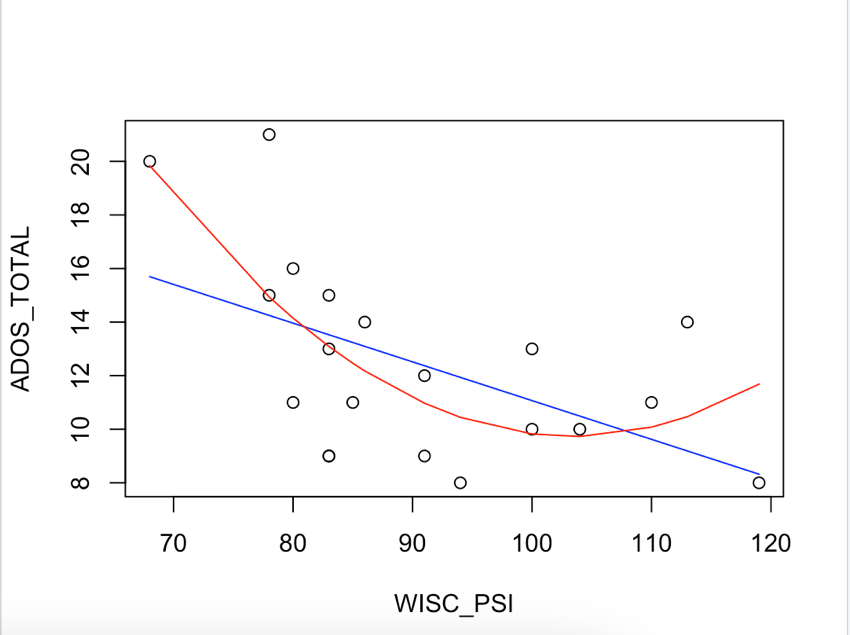


اگر برای این مدل نیز یک prediction انجام دهیم و آن را در غالب خط روی scatterplot دیتا اصلی بیاندازیم اینطوری می‌شود:





اگر هر دو خط را کنار هم روی scatterplot بیاندازیم اینطوری می‌شود:‌ 



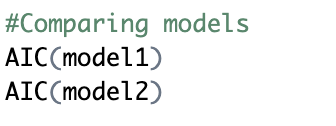
آیا اگر یک خط بالا پایین برود و همه نقاط data را کاور کند و دقیق به دیتا fit شده باشد، آیا مدل خوبی محسوب می‌شود؟



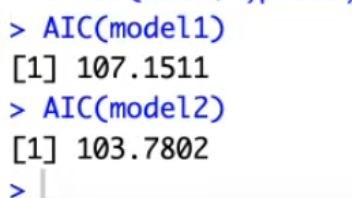
در اینجا مدل overfit شده است. مدل مثلا درجه ۲ خوب است ولی درجه ۵ دیگر زیادی است و به درد نمی‌خورد.

در overfitting فاصله نقطه‌ها از خط خیلی کم است ولی این مسئله general نیست. یعنی پیش‌بینی خوبی نمی‌تواند به ما بدهد. مدل‌های ساده پیش‌بینی‌های بهتری به ما می‌دهند. ما ترجیح می‌دهیم نقطه‌ها از خط پیش‌بینی شده فاصله داشته باشند ولی مدل ساده‌ باشد.

چطور می‌توانیم کیفیت مدل‌ها را با هم مقایسه کنیم؟ یک ارتباط ریاضی به اسم A I C وجود دارد که به صورت کمی می‌تواند مدل‌ها را مقایسه کند و هم overfit شدن را در نظر می‌گیرد و هم اینکه مدل به اندازه کافی به data فیت شده است یا نه. هر چه A I C کمتر باشد بهتر است.

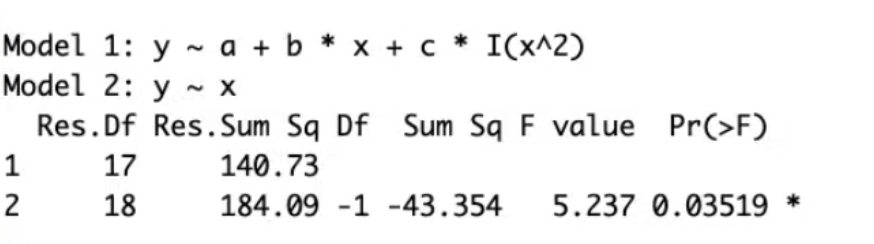


می‌بینیم که مدل ۲ از ۱ بهتر است.



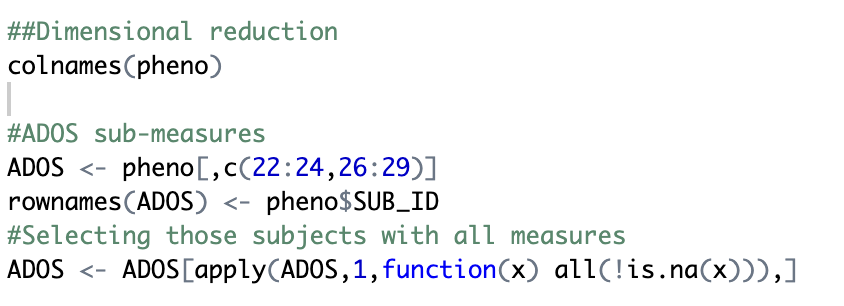
البته می‌توان بین دو مدل anova هم گرفت و آن‌ها را مقایسه کرد. چون می‌خواهیم بدانیم آیا واقعا رفتن از خطی به غیر خطی تفاوت معنادار آماری دارد و میارزد یا نه. یعنی برای اینکه بفهمیم آیا این بهتر شدن مدل به لحاظ آماری قابل توجه است یا نه را می‌توانیم با استفاده از anova مشخص کنیم.



که می‌بینیم معنادار می‌شود. اگر معنادار نمی‌شد ما از خطی به غیر خطی نمی‌رفتیم چون مدل خطی ساده‌تر است و باید برای ما توجیه آماری داشته باشد تا از خطی به غیر خطی برویم.

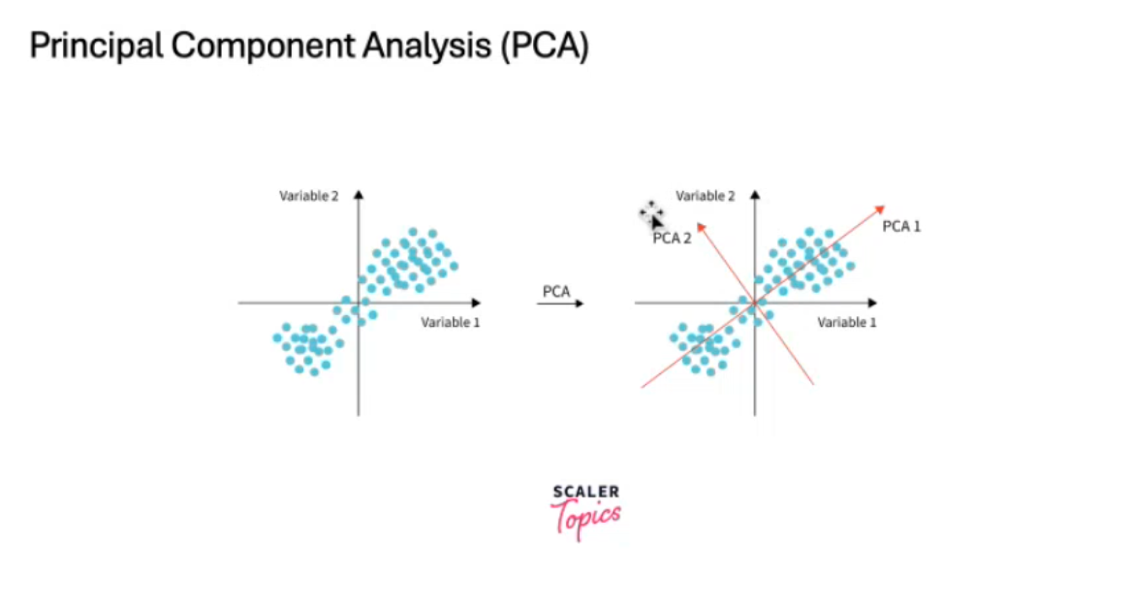
تمرین اول: برای دو متغیر processing speed (WISC-IV-PSI) و F I Qیک مدل خطی و یک مدل quadratic (غیر خطی) بزنید و ببینید کدام مدل بهتر است (کدام optimizeتر است).

در قسمت بعدی این جلسه می‌خواهیم به dimension reduction بپردازیم. باید کتابخانه lattice را اجرا کنیم. سپس یک subset از کل دیتا درست می‌کنیم که همه measureهای ADOS را داشته باشد (قبلش می‌رویم اسم ستون‌های دیتا را نگاه می‌کنیم و می‌بینیم که کدام ستون‌ها را باید انتخاب کنیم). سپس برای ردیف‌ها اسم سابجکت‌ها را قرار می‌دهیم و آن سابجکت‌هایی را انتخاب می‌کنیم که همه متغیرها را دارند و هیچ کدام از متغیرهاشان N A نیست.



حالا یک دیتا داریم که به زبان ریاضی ۷ ویژگی یا بعد دارد (چون ۷ تا از ستون‌ها را انتخاب کردیم). نمی‌توان به راحتی با این ۷ تا بعد کار کرد. ما می‌خواهیم کاری کنیم که این ۷ بعد را کم کنیم و دیتا را به جای ۷ تا چیز، با ۱ چیز توصیف کنیم. دو راهکار برای کاهش بعد وجود دارد: PCA و ICA.

در آنالیز PCA ما محورهای x و y را می‌‌چرخانیم و دیتا را در یک فضای جدید توصیف می‌کنیم. یعنی خود دیتا عوض نمی‌شود، ولی ما آن را در یک فضای جدید می‌بریم و توصیف می‌کنیم. قبلا اسم محورها variable 1 و variable 2 بود و حالا که محورها چرخیده‌اند اسمشان می‌شود PCA1 و PCA2.



حالا که فضا را چرخانده‌ایم همه چیز در راستای PCA1 است و انگار PCA2 دیگر اهمیتی ندارد. یعنی می‌توانیم از PCA2 صرف نظر کنیم. بنابراین به نظر می‌رسد که ما طوری فضا را چرخانده‌ایم که یک بعد از بقیه پررنگ‌تر شده است و می‌توانیم دیتا را با استفاده از همین بعد توصیف کنیم. پراکندگی دیتا حالا بیشتر در جهت PCA1 است و در جهت PCA2 پراکندگی نداریم.

البته این مثال ساده شده برای زمانی بود که ما ۲ بعد داشتیم و می‌خواستیم آن را کم کنیم و یک بعدش کنیم تا دیگر مجبور نباشیم برای گزارش یک داده هم x و هم y ش را گزارش کنیم و فقط بتوانیم با گزارش یک بعد کارمان را راه بیاندازیم.

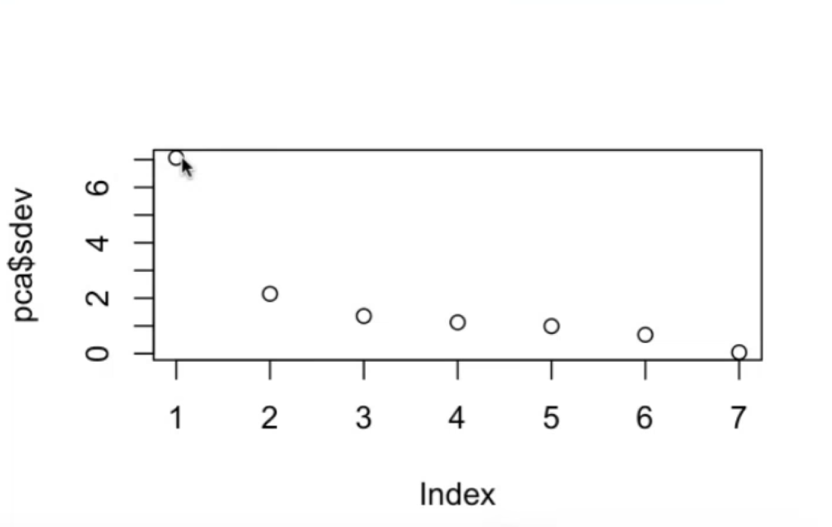
دو تا چیز در PCA مهم است:

1. یکی میزان چرخش (rotation)ی است که صورت گرفته . این چرخش یک ماتریکس متناظر دارد که به ما می‌گوید هر کدام از PCAها چقدرش از محور x گرفته شده و چقدرش از محور y. یعنی چه ترکیب‌ خطی از محور‌های قبلی ما معادل محورهای جدید ما می‌باشد. این چرخش در قالب یک ماتریکس چرخش به ما خروجی داده می‌شود.
2. دیتا ما در فضای جدید PCA. دیتا ما حالا در فضای جدید چگونه توصیف می‌شود.

حالا در R به این شکل PCA را می‌سازیم: (اول dimensionها را گرفتیم تا ببینیم چند بعد دارد.)

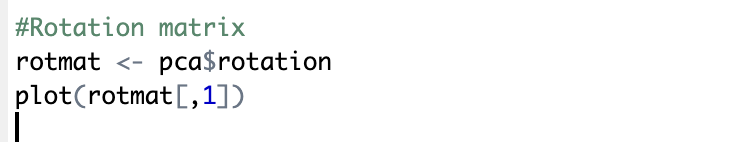


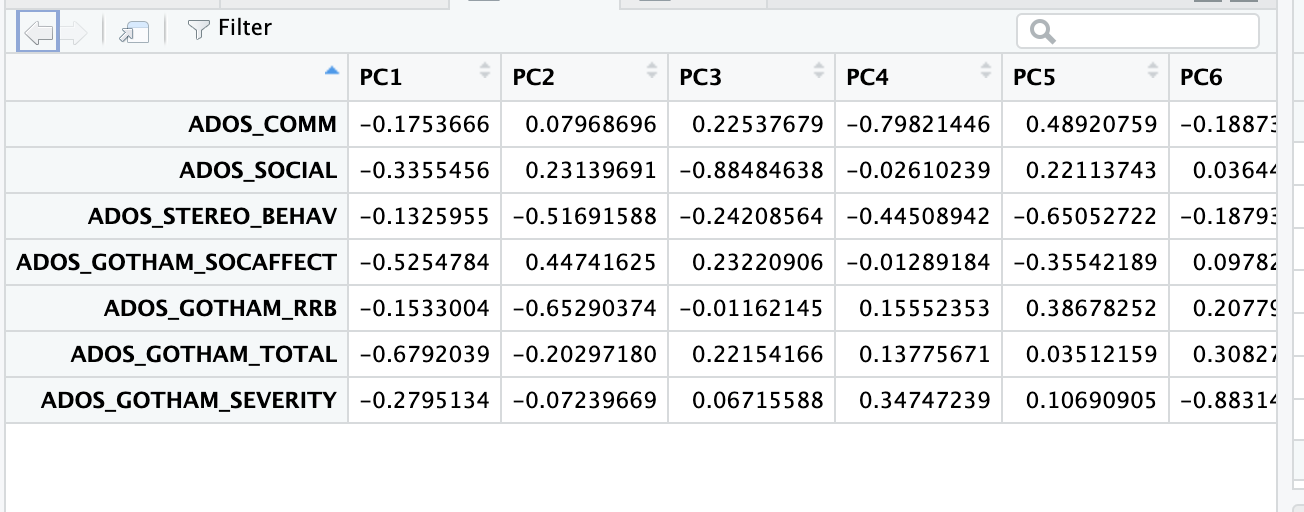
حالا pca را ساختیم. چند تا خروجی وجود دارد که به ما می‌گوید کیفیت این pca که ساختیم چطور است. یکی از آنها pca$sdev است که اگر plotش کنیم، چنین خروجی به ما می‌دهد:



حالا دیتا در فضای PCA چرخیده است واین نمودار می‌گوید که هر کدام از PCها دیتا ما را تا چه درصدی توصیف می‌کنند. همیشه طوری تنظیم می‌شود که PC1 از همه بالاتر باشد و به مرور کم شود. البته در واقع یک PCA خوب آنی است که ۱ش از همه بالاتر باشد و به شکل نمایی کم شود. یعنی ۲ نسبت به ۱ خیلی کمتر و ۳ نسبت به ۲ خیلی کمتر و ... باشد. این PCA که به آن در اینجا رسیدیم هم PC خوبی است. می‌بینیم که PCA1 به طرز قابل توجهی بیشتر از بقیه دیتا را توصیف می‌کند.

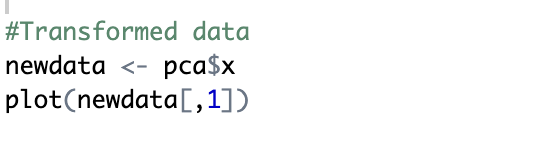
اگر بخواهیم ماتریکس چرخش آن را به دست آوریم باید این دستور را وارد کنیم:



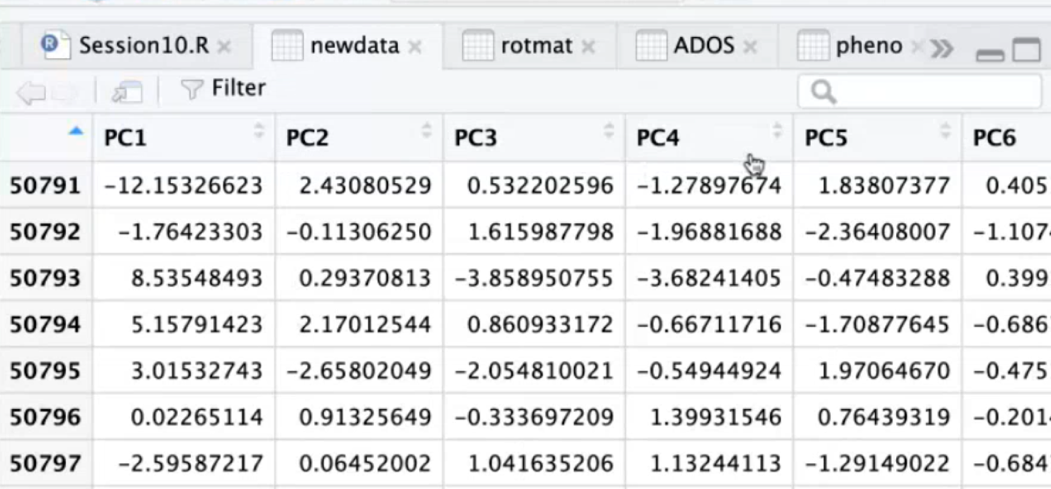


در واقع rotmat به ما می‌گوید که هر کدام از PC ها از کدام variableهای واقعی نشات می‌گیرند. و مثلا ما می‌توانیم بفهمیم در PC1 که انقدر از بقیه بهتر است کدام متغیر بوده که تاثیرگذارتر از بقیه بوده (در اینجا ADOS\_Gotham\_TOTAL) بوده است.

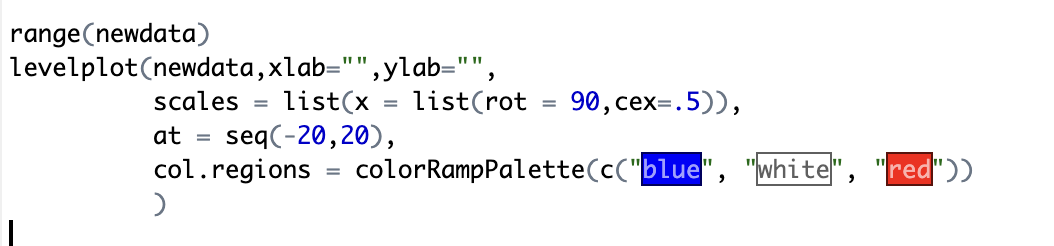
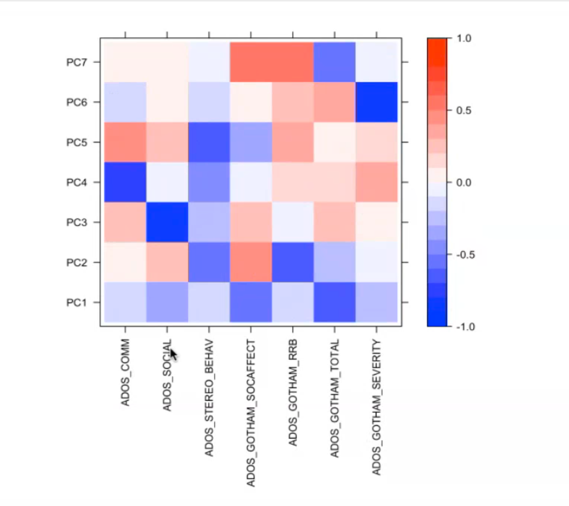
یه خروجی دیگه PCA دیتا ما در فضای جدید است. که با این دستور به دست می‌آوریم:

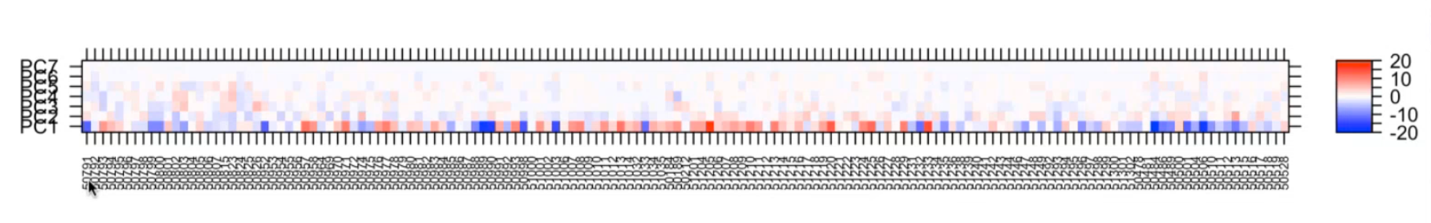


به ما scatterplot دیتا در فضای جدید را می‌دهد که می‌توان مشاهده کرد که PC1, PC2و... برای هر سابجکت چقدر است.



ما با دستور levelplot می‌توانیم یک hit map از rotation matrix داشته باشیم. این نمودار چه اطلاعاتی به ما می‌دهد؟ به ما می‌گوید که در هر PC چه ستون‌هایی مهم‌تر بوده‌اند. حالا اگر رنج را بین -۲۰ تا ۲۰ تنظیم کنیم (چون قبلش رنج دیتا جدید را گرفتیم و دیدیم از ۱۸- تا ۱۸ بود)، کل دیتاها را به ما می‌دهد.





در نمودار بالا هر چه رنگ به سمت سفید می‌رود، ارزش PC کمتر می‌شود.

تمرین ۲ و ۳:

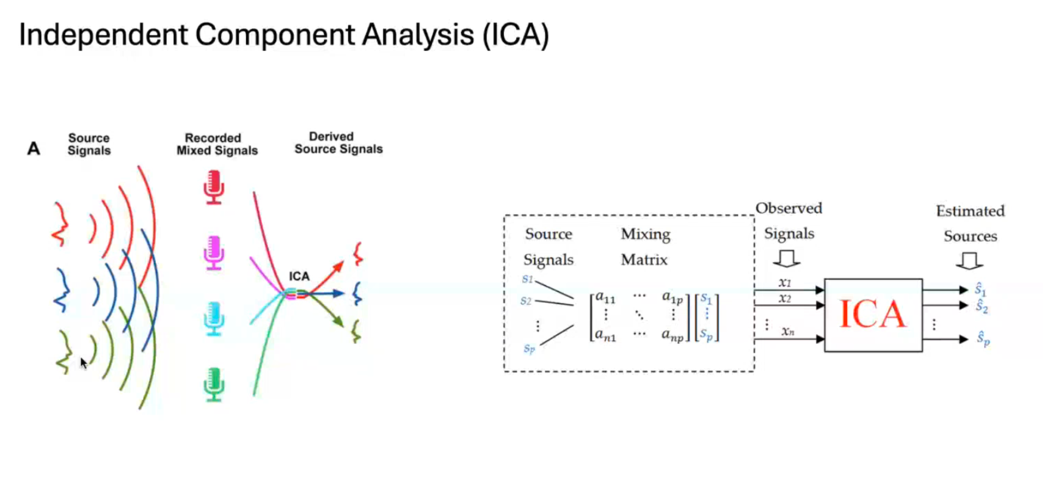
در دستور PCA دو اپشن center و scale داریم که می‌توان آن‌ها را T یا F گذاشت. به صورت دیفالت center=T است و scale=F. در تمرین ۲ یکی از این‌ها را activate کنید و در تمرین ۳ آن یکی را deactivate کنید و ببینید چه اتفاقی می‌افتد. نمودار PCA عوض خواهد شد و محورها تغییر می‌کنند. با استفاده از دستور sdev و میزان نمایی بودن نمودار آن تشخیص دهید که کدام نمودار بهتر است. هر چقدر نمودار سریع‌تر افت کند، بهتر است. (دستور کمی برای بررسی کیفیت PCA نداریم و به صورت کیفی باید آن را مشاهده کرد).

یک ابزار مهم دیگر که نوروساینس کاربرد دارد ICA است که معمولا برای پردازش سیگنال از آن استفاده می‌شود اما در اینجا ما برای dimensional reduction از آن استفاده می‌کنیم.

به لحاظ مفهومی ICA یک source estimation method است. اگر چند نفر در حال صحبت کردن باشند، ICA صدای همه را می‌گیرد و مشخص می‌کند که هر صدا متعلق به کدام فرد است. یعنی اگر یک دیتا کلی داشته باشیم و ندانیم منبع اجزای تشکیل دهنده این دیتا چه چیزهایی است، ICA می‌تواند سورس‌ها را تخمین بزند.

این می‌تواند از جهتی شبیه به dimensional reduction باشد چون در آن هم ما می‌خواهیم یک سری ابعاد جدید و بهتر برای دیتایی پیدا کنیم که یک مجموعه کلی می‌باشد. در اینجا هم ما میایم سورس‌های جدید و محورهای جدید تخمین می‌زنیم که داده‌ها را به شکل متفاوتی توصیف کنند. البته این روش از این جهت با PCA فرق دارد که در PCA محورها نسبت به هم اولویت بندی داشتند، یعنی PC1 در توصیف داده بهتر از PC2، و PC2 بهتر از PC1 بود. در اینجا ما باید به ICA بگوییم که چند محور جدید می‌خواهیم تا آن‌ها را به ما بدهد.

در اینجا mixing matrix شبیه همان rotation matrix است و estimated sources در واقع داده‌های ما در فضای جدید است (شبیه به new data در PCA).

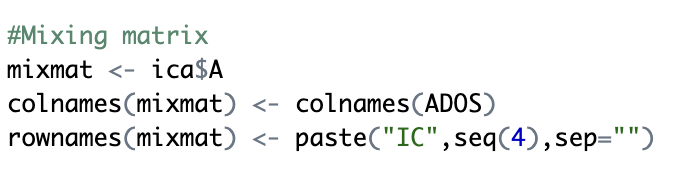


برای استفاده از ICA باید کتابخانه fastICA را اجرا کنیم.

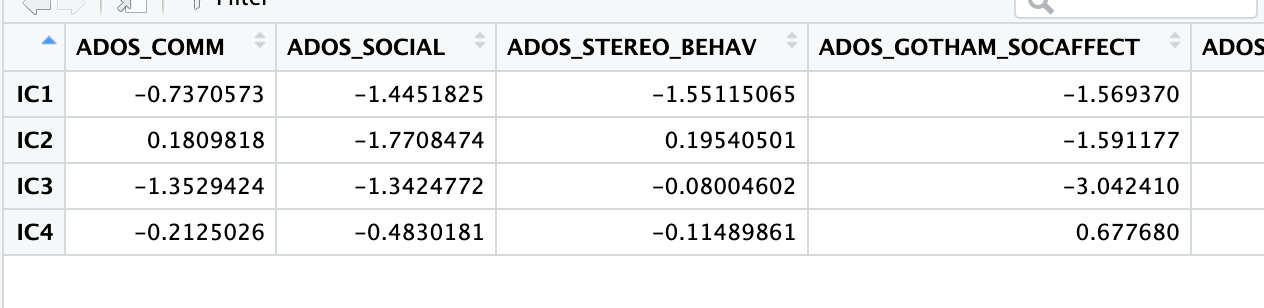


بعد این دستور را اجرا می‌کنیم:

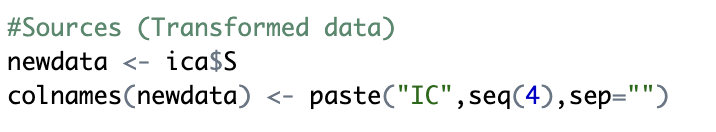
برای دیدن mixing matrix باید این دستور را اجرا کنیم. البته دو خط آخر آن برای این است که ردیف‌ها و ستون‌ها اسم داشته باشند. توجه کنید که در ICA هیچ‌کدام از IC ها نسبت به دیگری برتری ندارند و نمی‌شود گفت IC1 از IC2 بهتر است.

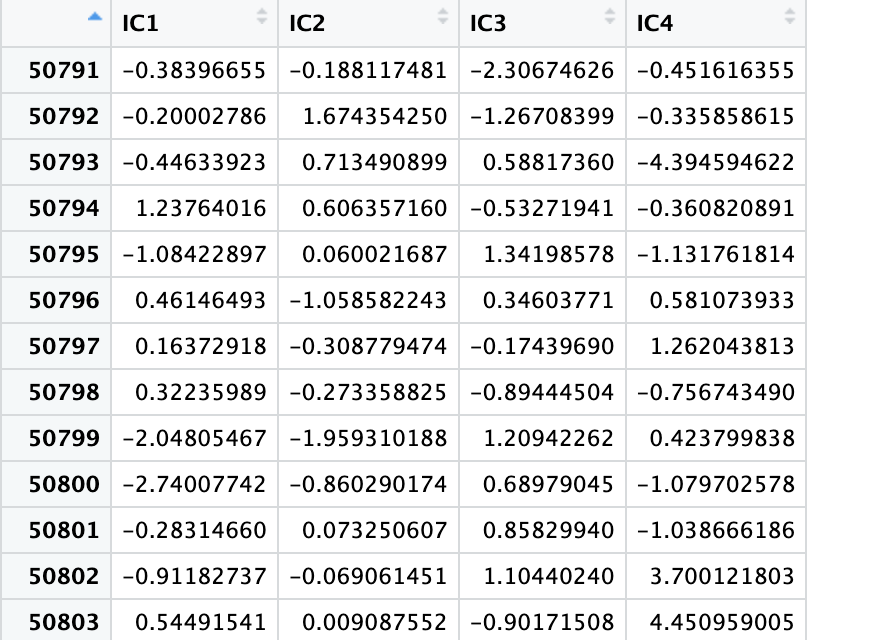


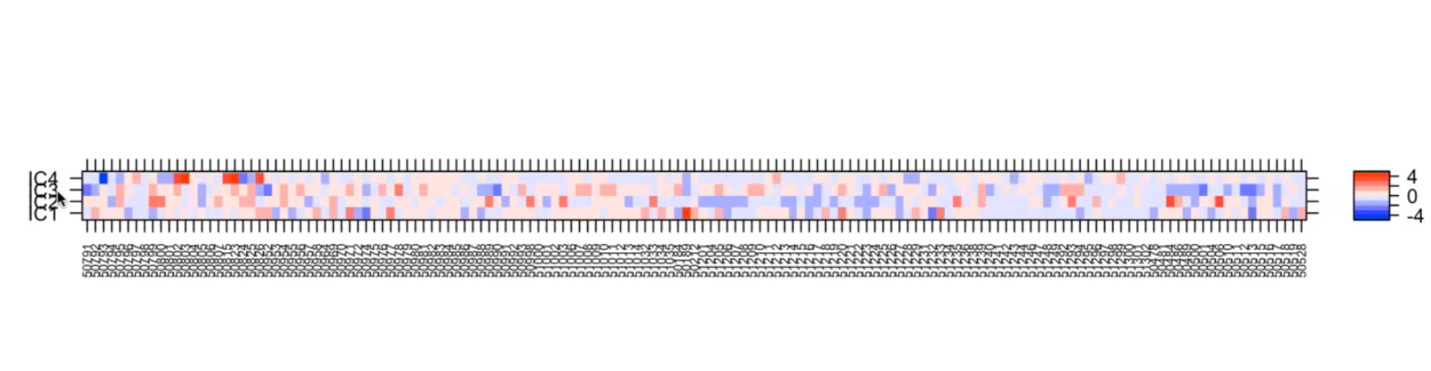
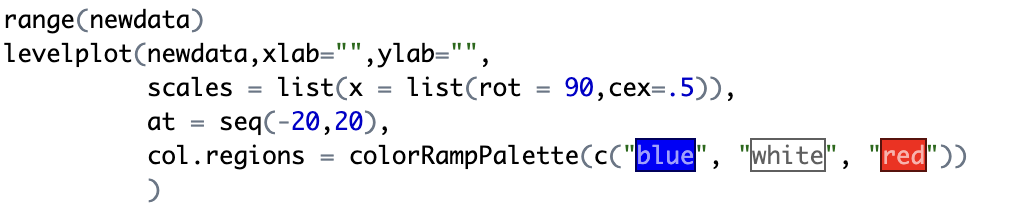
این mixing matrix است:



برای دیدن دیتا در فضای جدید هم از این دستور استفاده می‌کنیم:



دیتا در فضای جدید به این شکل است:

می‌توانیم مثل PCA برای آن levelplot یا hit map هم بکشیم اما مثل PCA نیست که PC های آخری دیگر به سمت سفیدی پیش بروند و بی‌ارزش باشند چون هیچ کدام از محورها نسبت به دیگری برتری ندارند. 

تمرین چهارم:

تعداد componentها را در ICA یک دفعه ۲ بذارید، یک دفعه ۳ و یک دفعه ۴. برای هر کدام یک mixing matrix و levelplot بکشید و توی پی‌دی‌اف بذارید و استدلال کنید که به لحاظ ریاضی کدام حالت ICA برای ما بهینه‌تر است و چه تعداد component بهتر است.

قسمت machine learning، سپیده کاردان

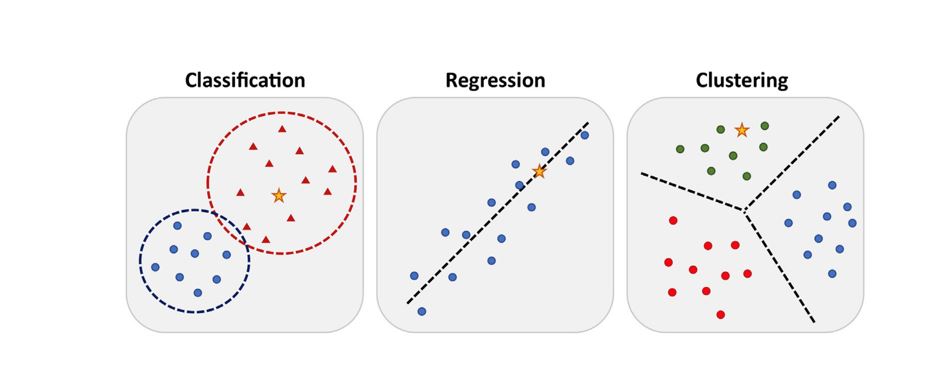
ماشین لرنینگ شامل دو دسته ی کلی می باشد:

1. Supervised: اگر یادگیری مدل بر اساس یک تارگت بیرونی باشد.

* Classification: اگر هدف دسته بندی داده ها مثلا بر اساس سالم یا بیمار بودن باشد(متغیر nominal).
* Regression: اگر هدف پیش بینی براساس یک متغیر پیوسته مانند سن باشد .

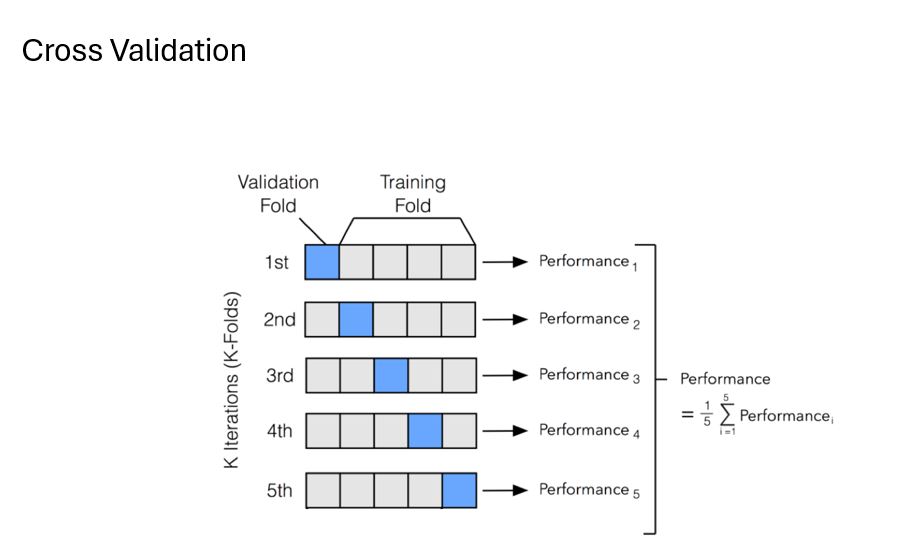
1. Unsupervised: اگر یادگیری تارگت بیرونی نداشته باشد. ما برای ماشین ویژگی را تعیین نمی کنیم و خودش به ما پیشنهاد می‏دهد.

* Clustering
* Dimension Reduction



برای حالت supervised اگر 100 داده داشته باشیم، بر اساس 80 داده مدل ساخته می‏شود و روی 20 داده‏ی بعدی تست می‏شود تا کیفیت مدل بررسی شود.

ممکن است 20 داده ای که برای تست انتخاب شده است، 20 داده ی خوبی نباشد یا به عبارتی reliable نباشد. به همین دلیل از cross validation استفاده می‏شود. در این روش به عنوان مثال 100 داده را به fold های مختلف تقسیم بندی می‏کنیم و هر دفعه تست را روی یک fold انجام می‏دهیم. هر بار تست بر روی یک fold مشخص به ما یک performance می‏دهد که نشان‏گر این است که چقدر مدل خوب کار می‏کند. در نهایت میانگین تمام performance ‏ها را گرفته و به آن cross validation گفته می‏شود.



**Machine learning: classification**

بر اساس کد زیر ADOS را انتخاب می‏کنیم:

ADOS <- pheno[,c(22:24,26:29)]

سپس یک dataframe می‏سازیم که در آن ADOS باشد و جنسیت در آن به عنوان یک factor باشد.

data <- data.frame(ADOS,as.factor(pheno$SEX) )

بعد از ساخت dataframe ، نام ستون آخر را به sex تغییر می‏دهیم. همچنین na ها را هم حذف می‏کنیم.

colnames(data)[8] <- "SEX"

data <- data[apply(data,1,function(x) all(!is.na(x))),]

هدف ما ساخت یک classifier است که بتواند براساس مقدار ADOS ، جنسیت را به ما بگوید.

برای انجام این کار و داشتن یک classifier که cross validation داشته باشد:

1. ساخت traincontrol ی که متد آن cross validation باشد و تعداد foldها را 10 قرار می‏دهیم. با این کار تعداد 149 داده را به 10 قسمت تقسیم می‏کند و هر دفعه تست را روی یکی از foldها انجام می‏دهد.

train\_control <- trainControl(method = "cv", number = 10)

1. سپس باید مشخص کنیم که چه چیزی را می‏خواهیم classify کنیم. در این جا می‏خواهیم جنسیت را بر اساس بقیه‏ی ستون‏ها classify کنیم. بقیه را با علامت نقطه در کد نشان می‏دهیم. همچنین باید الگوریتم ماشین لرنینگ را در این قسمت مشخص کنیم. الگوریتم‏های مختلف ماشین لرنینگ وجود دارد( این الگوریتم‏ها بسیار پیچیده هستند).

model <- train(SEX ~ ., data = data, method = "glmnet",trControl = train\_control)

1. بعد از train کردن مدل، مدل را پرینت می‏کنیم:

print(model)

1. Glmnet یک سری پارامتر دارد و به ازای هر پارامتر یک accuracy به ما می‏دهد. accuracy به ما می‏گوید که مدل چه‏ قدر خوب کار می‏کند. اگر برابر 1 باشد یعنی مدل می‏تواند به خوبی بر اساس ویژگی دسته‏بندی را انجام دهد و performance دسته‎بندی ما بیشتر است. kappa نیز بیان‏گر خطا می‏باشد که هر چه کم‏تر باشد بهتر است چرا که نشان‏دهنده‏ی این است که دسته‏بندی به خوبی انجام گرفته ایت.

glmnet

149 samples

7 predictor

2 classes: '1', '2'

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 133, 134, 134, 134, 135, 134, ...

Resampling results across tuning parameters:

alpha lambda Accuracy Kappa

0.10 6.005481e-05 0.8869048 0

0.10 6.005481e-04 0.8869048 0

0.10 6.005481e-03 0.8869048 0

0.55 6.005481e-05 0.8869048 0

0.55 6.005481e-04 0.8869048 0

0.55 6.005481e-03 0.8869048 0

1.00 6.005481e-05 0.8869048 0

1.00 6.005481e-04 0.8869048 0

1.00 6.005481e-03 0.8869048 0

Accuracy was used to select the optimal model using the

largest value.

The final values used for the model were alpha = 0.1 and lambda

= 0.006005481.

1. این بار با یک الگوریتم دیگر به نام pls دسته‏بندی را انجام می‏دهیم:

model <- train(SEX ~ ., data = data, method = "pls",trControl = train\_control)

1. بعد از train کردن مدل، مدل را پرینت می‏کنیم:

print(model)

1. در این مدل هم accuracy تقریبا برابر با 80 درصد می‏باشد.

Partial Least Squares

149 samples

7 predictor

2 classes: '1', '2'

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 134, 134, 134, 135, 133, 134, ...

Resampling results across tuning parameters:

ncomp Accuracy Kappa

1 0.8865476 0

2 0.8865476 0

3 0.8865476 0

Accuracy was used to select the optimal model using the

largest value.

The final value used for the model was ncomp = 1.

1. این بار با یک الگوریتم دیگر به نام svmLinear دسته‏بندی را انجام می‏دهیم:

model <- train(SEX ~ ., data = data, method = "svmLinear",trControl = train\_control)

1. بعد از train کردن مدل، مدل را پرینت می‏کنیم و در این مدل هم accuracy تقریبا برابر با 80 درصد می‏باشد.

print(model)

Support Vector Machines with Linear Kernel

149 samples

7 predictor

2 classes: '1', '2'

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 134, 135, 134, 134, 134, 135, ...

Resampling results:

Accuracy Kappa

0.8865476 0

Tuning parameter 'C' was held constant at a value of 1

1. این بار با یک الگوریتم دیگر به نام rf دسته‏بندی را انجام می‏دهیم:

model <- train(SEX ~ ., data = data, method = "rf",trControl = train\_control)

1. بعد از train کردن مدل، مدل را پرینت می‏کنیم و در این مدل accuracy با mtry های مختلف تغییر می‏کند و بهترین آن حالتی است که mtry=2 است.

print(model)

Random Forest

149 samples

7 predictor

2 classes: '1', '2'

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 135, 134, 133, 134, 134, 135, ...

Resampling results across tuning parameters:

mtry Accuracy Kappa

2 0.8668452 -0.02653931

4 0.8472619 -0.05124631

7 0.8414286 -0.04720450

Accuracy was used to select the optimal model using the

largest value.

The final value used for the model was mtry = 2.

در لینک زیر تمام الگوریتم‏های مختلف برای ماشین لرنینگ قرار دارد:

<https://rdrr.io/cran/caret/man/models.html>

تمرین:

Assignment 5: Classify subjects based on DSM-IV diagnostics categories, plot performances of classifiers as a bar plot

برای حل این تمرین کافی است به جای sex در دستور مربوط به dataframe، DSM\_IV\_TR را قرار دهید.

**Machine learning: regression**

رگرسورها قرار است یک عدد پیوسته‏ای را پیش‏بینی کنند. به عنوان مثال می‏خواهیم بر اساس ADOSها سن را پیش‏بینی کنیم.

بر اساس کد زیر ADOS را انتخاب می‏کنیم:

ADOS <- pheno[,c(22:24,26:29)]

سپس یک dataframe می‏سازیم که در آن ADOS باشد و سن را هم به آن اضافه می‏کنیم:

data <- data.frame(ADOS,pheno$AGE\_AT\_SCAN )

بعد از ساخت dataframe ، نام ستون آخر را به AGE تغییر می‏دهیم. همچنین na ها را هم حذف می‏کنیم.

colnames(data)[8] <- "AGE"

> data <- data[apply(data,1,function(x) all(!is.na(x))),]

در قسمت قبلی که sex را با دستور as.factor به عنوان یک فاکتور(متغیر nominal) تغییر دادیم و خودبه‏خود متوجه شد که classification هست. در این قسمت هم به دلیل نبود دستور as.factor خودبه‏خود متوجه می‏شود که یک عدد است و روش مورد استفاده رگرسیون است.

در ادامه برای داشتن یک regressor که cross validation داشته باشد:

1. ساخت traincontrol ی که متد آن cross validation باشد و تعداد foldها را 10 قرار می‏دهیم. با این کار داده ها را به 10 قسمت تقسیم می‏کند و هر دفعه تست را روی یکی از foldها انجام می‏دهد.

train\_control <- trainControl(method = "cv", number = 10)

1. سپس باید مشخص کنیم که چه چیزی را می‏خواهیم پیش‏بینی کنیم. در این جا می‏خواهیم سن را بر اساس بقیه‏ی ستون‏ها پیش‏بینی کنیم. بقیه را با علامت نقطه در کد نشان می‏دهیم. همچنین باید الگوریتم ماشین لرنینگ را در این قسمت مشخص کنیم.

model <- train(AGE ~ ., data = data, method = "glmnet",trControl = train\_control)

1. بعد از train کردن مدل، مدل را پرینت می‏کنیم:

print(model)

1. در این جا نیز Glmnet یک سری پارامتر دارد و به ازای هر پارامتر این بار به جای accuracy که بیان‏گر performance بود، Rsquared داریم که بیان‏گر correlation بین دیتای موجود و پیش بینی شده به توان دو می‏باشد. هر چه این مقدار به 1 نزدیک تر باشد مدل رگرسیون بهتر کار می‏کند. در این جا MAE و RMSE بیان‏گر خطا می‏باشند. در این جا Rsquared تقریبا برابر با 0.1 می‏باشد که مقدار پایینی است و الگوریتم خوب کار نکرده است.

glmnet

149 samples

7 predictor

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 134, 134, 134, 133, 134, 135, ...

Resampling results across tuning parameters:

alpha lambda RMSE Rsquared MAE

0.10 0.001017628 2.936261 0.07050073 2.428599

0.10 0.010176281 2.933981 0.07057048 2.427792

0.10 0.101762810 2.922209 0.06881723 2.427879

0.55 0.001017628 2.936200 0.07046417 2.428652

0.55 0.010176281 2.932900 0.07059221 2.427956

0.55 0.101762810 2.917463 0.06682777 2.429076

1.00 0.001017628 2.936483 0.07041539 2.428828

1.00 0.010176281 2.933095 0.06985925 2.428738

1.00 0.101762810 2.906851 0.07210494 2.424472

RMSE was used to select the optimal model using the smallest value.

The final values used for the model were alpha = 1 and lambda

= 0.1017628.

1. این بار با یک الگوریتم دیگر به نام pls دسته‏بندی را انجام می‏دهیم:

model <- train(SEX ~ ., data = data, method = "pls",trControl = train\_control)

1. بعد از train کردن مدل، مدل را پرینت می‏کنیم:

print(model)

1. در این الگوریتم هم Rsquared تقریبا برابر با 0.1 می‏باشد که مقدار پایینی است و الگوریتم خوب کار نکرده است.

Partial Least Squares

149 samples

7 predictor

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 135, 135, 133, 133, 134, 134, ...

Resampling results across tuning parameters:

ncomp RMSE Rsquared MAE

1 2.935936 0.1240920 2.463607

2 2.877728 0.1165610 2.398924

3 2.969997 0.1036529 2.449202

RMSE was used to select the optimal model using the smallest value.

The final value used for the model was ncomp = 2.

1. این بار با یک الگوریتم دیگر به نام svmLinear دسته‏بندی را انجام می‏دهیم:

model <- train(SEX ~ ., data = data, method = "svmLinear",trControl = train\_control)

1. بعد از train کردن مدل، مدل را پرینت می‏کنیم و در این مدل هم Rsquared کوچک می‏باشد و الگوریتم خوب کار نکرده است.

print(model)

Support Vector Machines with Linear Kernel

149 samples

7 predictor

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 135, 136, 133, 134, 134, 133, ...

Resampling results:

RMSE Rsquared MAE

3.09209 0.04628984 2.519763

Tuning parameter 'C' was held constant at a value of 1

1. این بار با یک الگوریتم دیگر به نام rf دسته‏بندی را انجام می‏دهیم:

model <- train(SEX ~ ., data = data, method = "rf",trControl = train\_control)

1. بعد از train کردن مدل، مدل را پرینت می‏کنیم و در این مدل هم Rsquared کوچک می‏باشد و الگوریتم خوب کار نکرده است.

Random Forest

149 samples

7 predictor

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 134, 134, 134, 134, 134, 135, ...

Resampling results across tuning parameters:

mtry RMSE Rsquared MAE

2 2.945375 0.07032262 2.404820

4 2.967182 0.07290702 2.430476

7 2.952312 0.07989419 2.436141

RMSE was used to select the optimal model using the smallest value.

The final value used for the model was mtry = 2.

با توجه به این که هیچ یک از چهار مدل مناسب نبود می‏توان دو نتیجه گرفت:

1. الگوریتم انتخابی مناسب نبوده.
2. ویژگی‏های انتخاب شده بار اطلاعاتی کافی برای پیش‏بینی سن را ندارند(با توجه به این که الگوریتم‏های مختلف را استفاده کردیم، این گزینه محتمل‏تر است).

تمرین:

Assignment 6: Classify subjects based on social communication questionnaire, plot performances of classifiers as a bar plot

برای حل این تمرین کافی است به جای AGE در دستور مربوط به dataframe، SCQ\_TOTAL را قرار دهید.

**Machine learning: clustering**

این روش unsupervised بوده و برخلاف دو روش قبلی که supervised بودند، به آن تارگت نمی‏دهیم.

بر اساس کد زیر ADOS را انتخاب می‏کنیم و SUB\_ID را به عنوان اسم ردیف ها انتخاب می‏کنیم:

ADOS <- pheno[,c(22:24,26:29)]

rownames(data) <- pheno$SUB\_ID

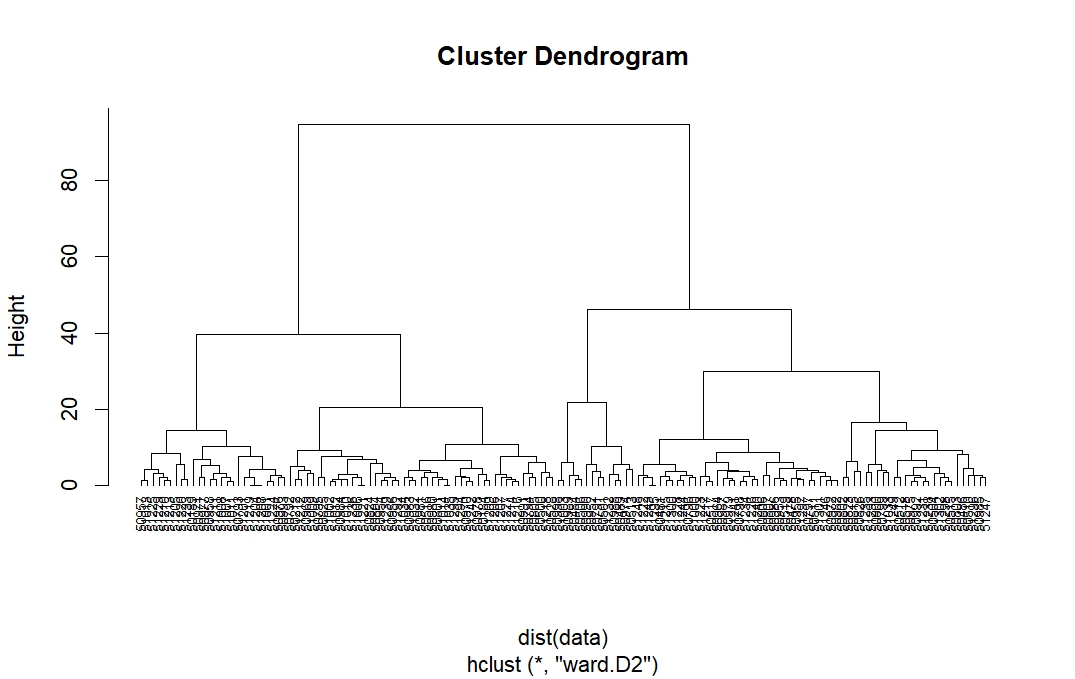
data <- data[apply(data,1,function(x) all(!is.na(x))),]

انتظار داریم که ماشین خودش برای ما بدون هیچ تارگتی یک خوشه بندی انجام دهد. روش‏های clustering متعدد می‏باشد. در این جا فقط به روش hierarchical اشاره می‏کنیم. روش hierarchical الگوریتم‏های مختلفی دارد. روش hierarchical بر این اساس عمل می‏کند که فاصله‏ی subjectها را دو به دو با هم مقایسه می‏کند و هم‏چنین بررسی می‏کند که چه‏قدر رفتارهایشان شبیه به هم هست. یعنی اگر به عنوان مثال 149 subject داشته باشیم یک ماتریکس 149 در 149 می‏سازد و ارتباط یعنی شباهت و تفاوت آن‏ها را دو به دو در موارد مختلف بررسی می‏کند.

برای hierarchical clustering ابتدا در کدی فواصل را با دستور dist حساب می‏کنیم و الگوریتم استفاده شده را نیز ذکر می‏کنیم:

hc <- hclust(dist(data), method = "ward.D2")

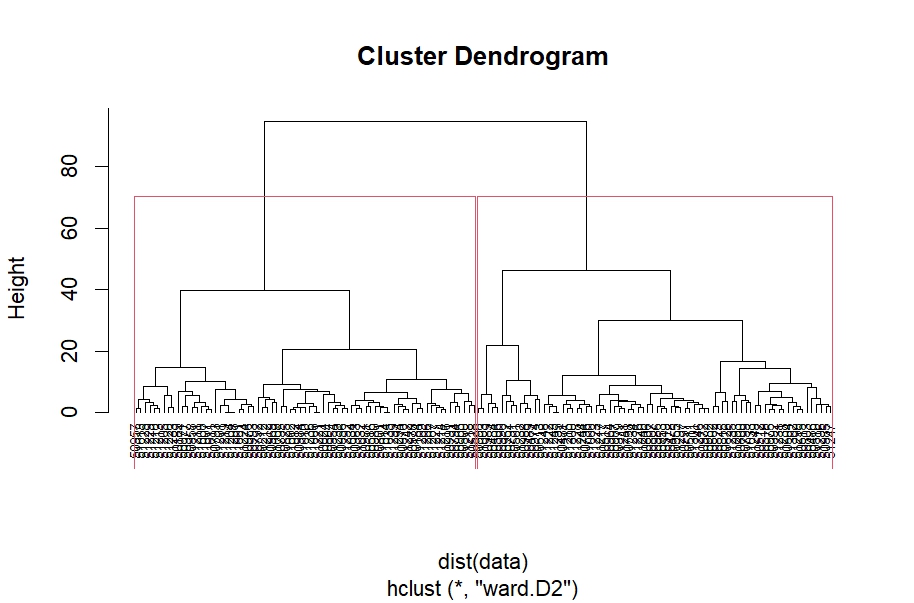
برای پلات کردن الگوریتم، از دستور زیر استفاده می‏کنیم:

plot(hc, cex = 0.6, hang = -1)

در تصویر بالا subject های مختلف را کلاستربندی کرده است.

برای توصیف بهتر پلات رسم شده می‏توان از دستور زیر استفاده کرد که در آن k را می‏توان برابر 2 قرار داد که نشان‏دهنده‏ی جایی است که دندروگرام به دسته کلاستر شده است(k را می‏توان برابر با اعداد مختلفی قرار داد):

rect.hclust(hc, k = 2)



با دستور زیر می‏توان مشخص کرد که هر subject در کدام کلاستر قرار می‏گیرد:

clusters <- cutree(hc, k = 2)

clusters

50791 50792 50793 50794 50795 50796 50797 50798 50799 50800 50801

1 2 2 2 2 1 1 1 1 1 2

50802 50803 50804 50805 50806 50807 50815 50823 50824 50825 50826

1 1 2 2 1 1 1 1 2 2 1

50952 50953 50954 50955 50956 50957 50958 50964 50969 50970 50971

1 1 2 1 2 2 2 1 2 2 2

50972 50974 50975 50976 50977 50978 50979 50980 50981 50982 50983

1 1 1 2 2 2 1 2 2 1 2

50984 50985 50986 50987 50988 50989 50990 50991 50992 50993 50998

2 1 1 1 1 1 1 2 1 2 1

51000 51001 51002 51003 51006 51007 51008 51009 51010 51011 51012

1 2 2 1 2 2 2 2 2 2 2

51013 51014 51032 51033 51034 51035 50184 50189 50212 51201 51204

2 2 2 1 2 2 2 2 2 2 2

51205 51206 51207 51208 51209 51210 51211 51212 51213 51214 51215

2 2 2 2 2 2 2 2 2 2 1

51216 51217 51218 51219 51220 51221 51222 51223 51224 51225 51226

2 1 2 2 2 2 1 1 2 2 1

51227 51228 51229 51231 51232 51233 51234 51235 51236 51238 51239

2 1 2 1 1 2 1 1 2 1 2

51240 51241 51242 51243 51244 51246 51247 51248 51249 51292 51293

1 1 2 1 1 1 1 1 1 2 1

51294 51295 51296 51297 51298 51299 51300 51301 51302 50478 50481

2 1 1 1 1 2 1 1 1 1 1

50484 50486 50489 50500 50501 50504 50506 50510 50511 50512 50513

1 1 1 2 1 1 1 1 1 1 1

50515 50516 50517 50518 50519 50528

1 2 1 1 1 2

با سرچ در قسمت help می‏توان سایر الگوریتم های مربوط به hclust(hierarchical) را پیدا کرد. هر کدام از این الگوریتم‏ها می‏تواند یک مذل خوشه‏بندی به ما ارائه دهد. اما سوال این است که کدام خوشه‏بندی بهتر است؟ برای همین نیاز به یک متر داریم به نام width silhouette. این معیار برای ما محاسبه می‏کند که یک subject چقدر نزدیک به هم کلاستری‏های خود و چه قدر دور از سایر کلاسترها می‏باشد. باید از sil\_width میانگین بگیریم که نشان‏دهنده‎ی کیفیت کلاسترینگ می‏باشد.

برای این کار از دستور زیر استفاده می‏کنیم:

sil <- silhouette(clusters, dist(data))

sil

cluster neighbor sil\_width

[1,] 1 2 0.50472671

[2,] 2 1 -0.20878598

[3,] 2 1 0.50619185

[4,] 2 1 0.51855541

[5,] 2 1 0.38756128

[6,] 1 2 0.02166602

[7,] 1 2 0.34532468

[8,] 1 2 0.36591633

[9,] 1 2 0.47835891

[10,] 1 2 0.45494266

[11,] 2 1 0.54367289

[12,] 1 2 0.29257530

[13,] 1 2 0.41031553

[14,] 2 1 0.12635995

[15,] 2 1 0.39432556

[16,] 1 2 0.44761424

[17,] 1 2 0.26434028

[18,] 1 2 0.17426843

[19,] 1 2 0.27009296

[20,] 2 1 0.18226948

[21,] 2 1 0.26173997

[22,] 1 2 0.18917939

[23,] 1 2 0.49385942

[24,] 1 2 0.28406575

[25,] 2 1 0.18106831

[26,] 1 2 0.43661896

[27,] 2 1 0.13396241

[28,] 2 1 0.55832233

[29,] 2 1 0.61567256

[30,] 1 2 0.35038570

[31,] 2 1 0.13947094

[32,] 2 1 0.53495795

[33,] 2 1 0.55383236

[34,] 1 2 0.37904304

[35,] 1 2 0.47399410

[36,] 1 2 0.50617504

[37,] 2 1 0.62792645

[38,] 2 1 0.57181817

[39,] 2 1 0.53586504

[40,] 1 2 0.28947901

[41,] 2 1 0.59938651

[42,] 2 1 0.52992033

[43,] 1 2 0.34838257

[44,] 2 1 0.61764094

[45,] 2 1 0.37415441

[46,] 1 2 0.53996150

[47,] 1 2 0.17236670

[48,] 1 2 0.21212668

[49,] 1 2 0.54802965

[50,] 1 2 0.42846164

[51,] 1 2 0.44517729

[52,] 2 1 0.58206397

[53,] 1 2 0.51602526

[54,] 2 1 0.62483537

[55,] 1 2 0.46891052

[56,] 1 2 0.02948915

[57,] 2 1 0.60756934

[58,] 2 1 0.45551948

[59,] 1 2 0.45738502

[60,] 2 1 0.34788941

[61,] 2 1 0.63011527

[62,] 2 1 0.61104439

[63,] 2 1 -0.25966416

[64,] 2 1 0.46731380

[65,] 2 1 0.62061309

[66,] 2 1 0.47993548

[67,] 2 1 0.56179300

[68,] 2 1 0.58590233

[69,] 2 1 0.60353087

[70,] 1 2 0.50610802

[71,] 2 1 0.60863042

[72,] 2 1 0.58590233

[73,] 2 1 0.57559006

[74,] 2 1 0.49689823

[75,] 2 1 -0.03131786

[76,] 2 1 0.61743097

[77,] 2 1 0.59795427

[78,] 2 1 0.46334251

[79,] 2 1 0.36038457

[80,] 2 1 0.55018329

[81,] 2 1 0.59795427

[82,] 2 1 0.59027922

[83,] 2 1 0.59491342

[84,] 2 1 0.63826340

[85,] 2 1 -0.06814166

[86,] 2 1 0.59760788

[87,] 2 1 0.49057885

[88,] 1 2 -0.19846372

[89,] 2 1 0.53087773

[90,] 1 2 0.30208181

[91,] 2 1 0.31395837

[92,] 2 1 0.59795427

[93,] 2 1 0.52679047

[94,] 2 1 0.34788941

[95,] 1 2 -0.08029288

[96,] 1 2 0.34645390

[97,] 2 1 0.60679229

[98,] 2 1 0.56115975

[99,] 1 2 0.35472539

[100,] 2 1 0.45626318

[101,] 1 2 -0.17591354

[102,] 2 1 0.59015959

[103,] 1 2 0.46861418

[104,] 1 2 0.51185022

[105,] 2 1 0.51191599

[106,] 1 2 0.35344251

[107,] 1 2 0.47094639

[108,] 2 1 0.43221208

[109,] 1 2 0.49319388

[110,] 2 1 0.18432408

[111,] 1 2 0.31608174

[112,] 1 2 0.27316803

[113,] 2 1 0.52976751

[114,] 1 2 0.02602001

[115,] 1 2 -0.24998439

[116,] 1 2 0.41145743

[117,] 1 2 0.56471975

[118,] 1 2 -0.02465471

[119,] 1 2 0.49589126

[120,] 2 1 0.08321955

[121,] 1 2 0.42985262

[122,] 2 1 0.54814891

[123,] 1 2 0.02602001

[124,] 1 2 0.54055458

[125,] 1 2 0.45077151

[126,] 1 2 0.54551312

[127,] 2 1 0.48406091

[128,] 1 2 -0.13011564

[129,] 1 2 0.41200188

[130,] 1 2 0.51760485

[131,] 1 2 0.46606105

[132,] 1 2 0.01917509

[133,] 1 2 0.39842363

[134,] 1 2 0.52223279

[135,] 1 2 0.52494079

[136,] 2 1 0.42086392

[137,] 1 2 0.50847014

[138,] 1 2 0.52937084

[139,] 1 2 0.41318928

[140,] 1 2 0.55255878

[141,] 1 2 0.42323632

[142,] 1 2 0.50626271

[143,] 1 2 0.43869000

[144,] 1 2 0.53894347

[145,] 2 1 0.50031076

[146,] 1 2 0.32550182

[147,] 1 2 0.44798385

[148,] 1 2 0.30673718

[149,] 2 1 0.43563210

attr(,"Ordered")

[1] FALSE

attr(,"call")

silhouette.default(x = clusters, dist = dist(data))

attr(,"class")

[1] "silhouette"

سپس میانگین sil\_width را محاسبه می‏کنیم:

mean(sil[,3])

[1] 0.4000791

تمرین:

Assignment 7: Run clustering with different methods and K=2-3,make a table containing silhouette means plot the best clustering with the largest silhouette mean qualitatively explore any relevance between clustering labels and phenotypical data

برای حل این تمرین باید از الگوریتم‏های مختلف hclust استفاده کنیم و k را برابر با 2 و 3 قرار دهیم و میانگین sil\_width را حساب کنیم و یک جدول بسازیم و بهترین الگوریتم را پیدا کنیم.