

An Introduction to *R*

Part 4: Covariance Structures in Mixed Models and Dimension Reduction and Classification

Kjell Johnson



From Part 3:

Repeated Measures Example

- Polysomnographic analysis of rat cortical EEG - wakefulness
 - 16 animals were randomly selected
 - 8 were randomly treated with vehicle; remaining 8 received Drug_A
 - Total time awake post treatment (0-3 hr) was measured for each animal at days 0, 1, 2, 4, 5, 8, 11, and 14

Data

Animal	Day	Drug	TTA
1	0	vehicle	27.0
1	1	vehicle	43.9
1	2	vehicle	40.2
1	4	vehicle	69.7
1	5	vehicle	52.4
1	8	vehicle	37.1
1	11	vehicle	37.5
1	14	vehicle	73.2
2	0	vehicle	54.2
...
15	8	DrugA	84.4
15	11	DrugA	60.9
15	14	DrugA	97.8
16	0	DrugA	65.5
16	1	DrugA	71.9
16	2	DrugA	68.8
16	4	DrugA	92.3
16	5	DrugA	81.5
16	8	DrugA	88.4
16	11	DrugA	71.5
16	14	DrugA	92.8

Statistical Model

$$Y_{i,j,k} = \mu_{\dots} + \alpha_i + \rho_{i,j} + \tau_k + (\alpha\tau)_{i,k} + \epsilon_{i,j,k}$$

- $Y_{i,j,k}$ is the observed response value for the j^{th} subject on the i^{th} drug at time k .
 - $i = 1, \dots, \# \text{ of drugs (2)}$
 - $j = 1, \dots, \# \text{ of subjects within drug (8)}$
 - $k = 1, \dots, \# \text{ of timepoints (8)}$
- μ_{\dots} is the overall average response
- $\rho_{i,j}$ is the effect corresponding to the j^{th} subject on the i^{th} drug and is random and follows a normal distribution [random]
- τ_k is the effect at time k [fixed]
- $(\alpha\tau)_{i,k}$ is the interaction between the i^{th} drug at time k [fixed]
- $\epsilon_{i,j,k}$ is the random variation of the j^{th} subject not explained by drug i at time k

Covariance Structure

- The $\rho_{i,j}$ term is used to model variation within experimental units
- Lots of covariance structures exist to model this variation. Here are a few common types:
 - Compound symmetry: All measurements within a subject are equally correlated with each other
 - Autoregressive: Correlation within subjects is less the further the measurements are apart in time
 - Symmetric: A general symmetric structure

How to Analyze in R?

Perform mixed model analysis:

```
wakeMixed <- lmer(TTA ~ Drug + (1|Drug/Animal) +  
                  Day + Drug:Day, data=wake)
```

But lmer does not allow us to specify the correlation for random effects

Mixed models with nlme:

```
install.packages("nlme")  
library(nlme)
```

Try AR1 and Symmetric correlation models:

```
wakeMixedAR1 <- lme(TTA ~ Drug + Day + Drug:Day,  
                    random=~1|Drug/Animal,  
                    data=wake,  
                    correlation=corAR1())
```

```
wakeMixedSymm <- lme(TTA ~ Drug + Day + Drug:Day,  
                     random=~1|Drug/Animal,  
                     data=wake,  
                     correlation=corSymm())
```

Compare the Two Models

```
anova(wakeMixedAR1, wakeMixedSymm)
```

	Model	df	AIC	BIC	logLik	Test	L.Ratio	p-value
wakeMixedAR1	1	20	875.1559	929.5259	-417.5780			
wakeMixedSymm	2	47	902.2501	1030.0196	-404.1251	1 vs 2	26.90583	0.4689

In this case, the covariance matrices do not produce significantly different fits for this data.

Overview

- General dimension reduction
 - Principal components analysis (PCA)
- Classification methods
 - Partial least squares
 - Recursive partitioning
 - Random forests

Why Dimension Reduction?

- The data may have many more variables than samples
 - -omics data
 - Imaging data
 - Time-course data
- The variables may be highly correlated (redundant), and we wish to retain uncorrelated variables
- We wish to see a 2- or 3-dimensional visualization of the samples


What is PCA?

- PCA finds linear combinations of the original variables that summarize a maximum amount of variation in the original data

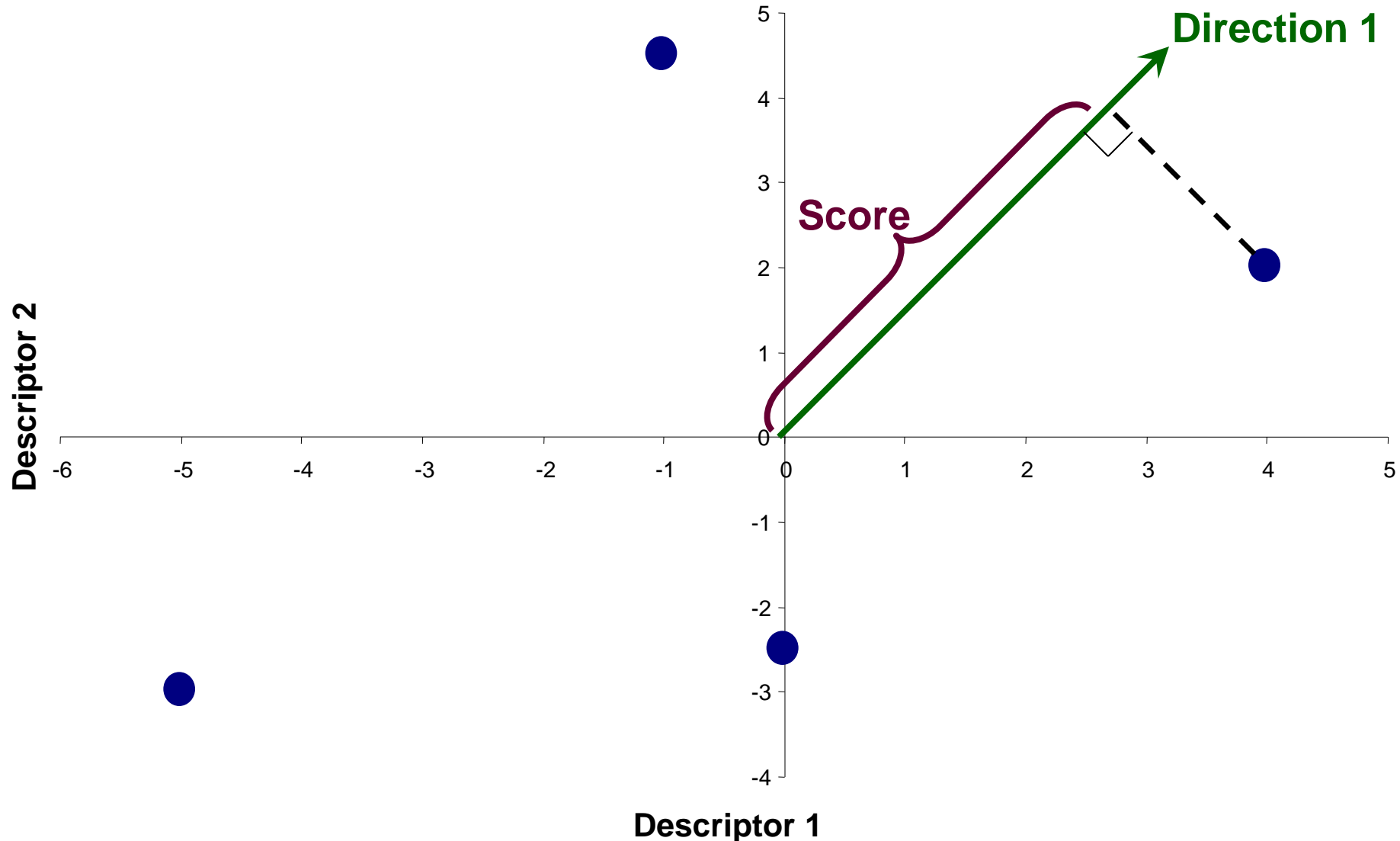
$$a_1X_1 + a_2X_2 + \cdots + a_pX_p$$

- PCA lingo:
 - Score: the linear combination of the original variables.
 - Direction: the vector that points towards maximum variation

Facts

- PCA seeks variability. Therefore if variables have much different scales, PCA will focus on variables with larger variation. 
 - Variables should be centered and scaled prior to PCA
- We can derive as many components as there are variables
- The scores from each component are *uncorrelated* with the scores from every other component. This implies that each component summarize a *unique* aspect of the original data.

Visualizing Principal Components Analysis (PCA)



Example Data

- Beta time-course, measured every 30 minutes to 22 hours:

Animal	t01	t02	t03	t04	t05	t06	...	t42	t43	t44
M01	0.12	0.11	0.11	0.11	0.12	0.12	...	0.09	0.09	0.10
M02	0.10	0.09	0.10	0.10	0.09	0.09	...	0.07	0.07	0.08
M03	0.10	0.10	0.09	0.09	0.09	0.09	...	0.08	0.08	0.07
M04	0.13	0.11	0.12	0.11	0.12	0.12	...	0.12	0.10	0.10
M05	0.10	0.10	0.09	0.10	0.10	0.09	...	0.08	0.07	0.07
M06	0.14	0.13	0.13	0.13	0.12	0.13	...	0.11	0.11	0.11
M07	0.12	0.12	0.12	0.12	0.12	0.12	...	0.11	0.10	0.11
M08	0.11	0.11	0.10	0.10	0.11	0.10	...	0.08	0.09	0.09
M09	0.11	0.12	0.10	0.11	0.11	0.10	...	0.09	0.10	0.08
M10	0.09	0.09	0.09	0.09	0.08	0.09	...	0.11	0.10	0.11
M11	0.04	0.06	0.08	0.07	0.03	0.06	...	0.04	0.03	0.07
M12	0.04	0.05	0.10	0.08	0.07	0.09	...	0.10	0.10	0.11
M13	0.10	0.10	0.08	0.09	0.09	0.09	...	0.10	0.10	0.09
M14	0.09	0.10	0.09	0.09	0.09	0.09	...	0.11	0.08	0.08
M15	0.13	0.14	0.12	0.12	0.12	0.11	...	0.11	0.11	0.10

Bring the Data into R & Perform PCA on Centered and Scaled Data

Set the working directory:

```
myLocation <- "c:/Part4"  
setwd(myLocation)
```

Get data:

```
beta <- read.csv("beta.csv", header=TRUE)
```

Partition animal and variables into separate objects

```
betaAnimal <- subset(beta, select=Animal)  
betaVars <- subset(beta, select=-Animal)
```

Perform PCA on centered and scaled variables

```
pcaBeta <- prcomp(x=betaVars, center=TRUE, scale.=TRUE,  
                  newdata=betaVars)
```

```
names(pcaBeta)
```

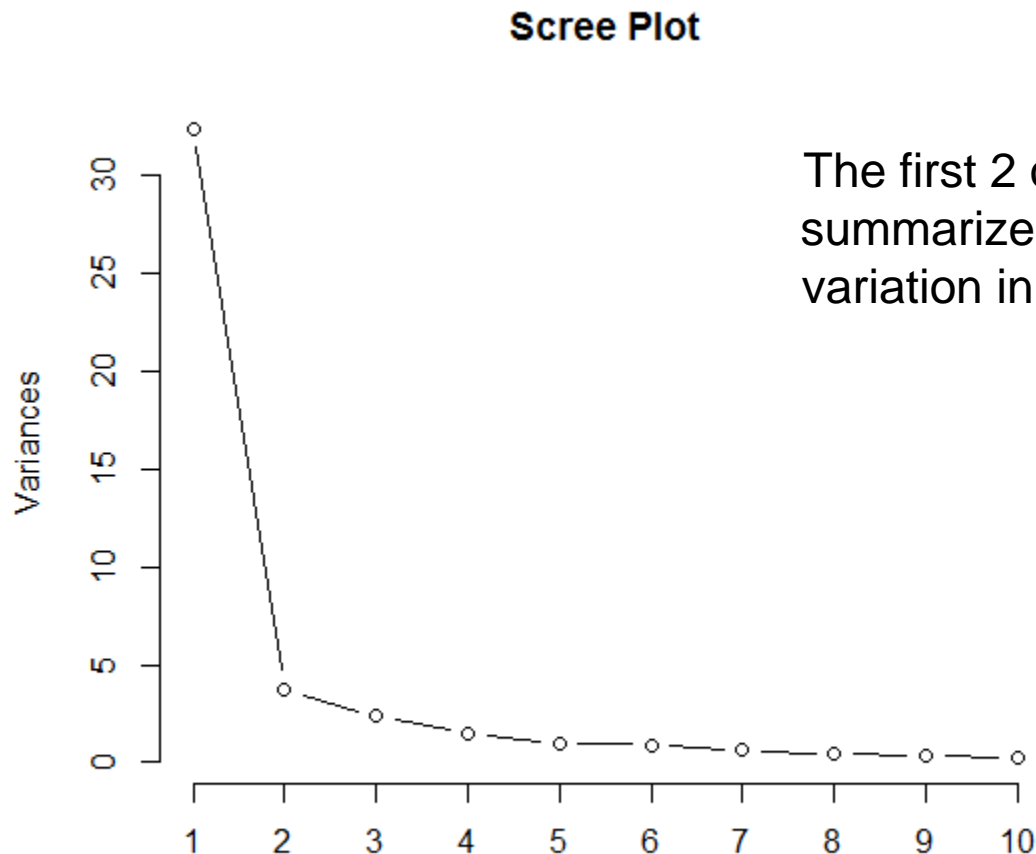
```
[1] "sdev"      "rotation" "center"   "scale"    "x"
```



Scores

Scree Plot: Variance Summarized by Each Component

```
screeplot(pcaBeta, type="lines",  
          main="Scree Plot")
```



The first 2 or 3 components summarize a majority of the variation in the original data

Visualize First Two Components

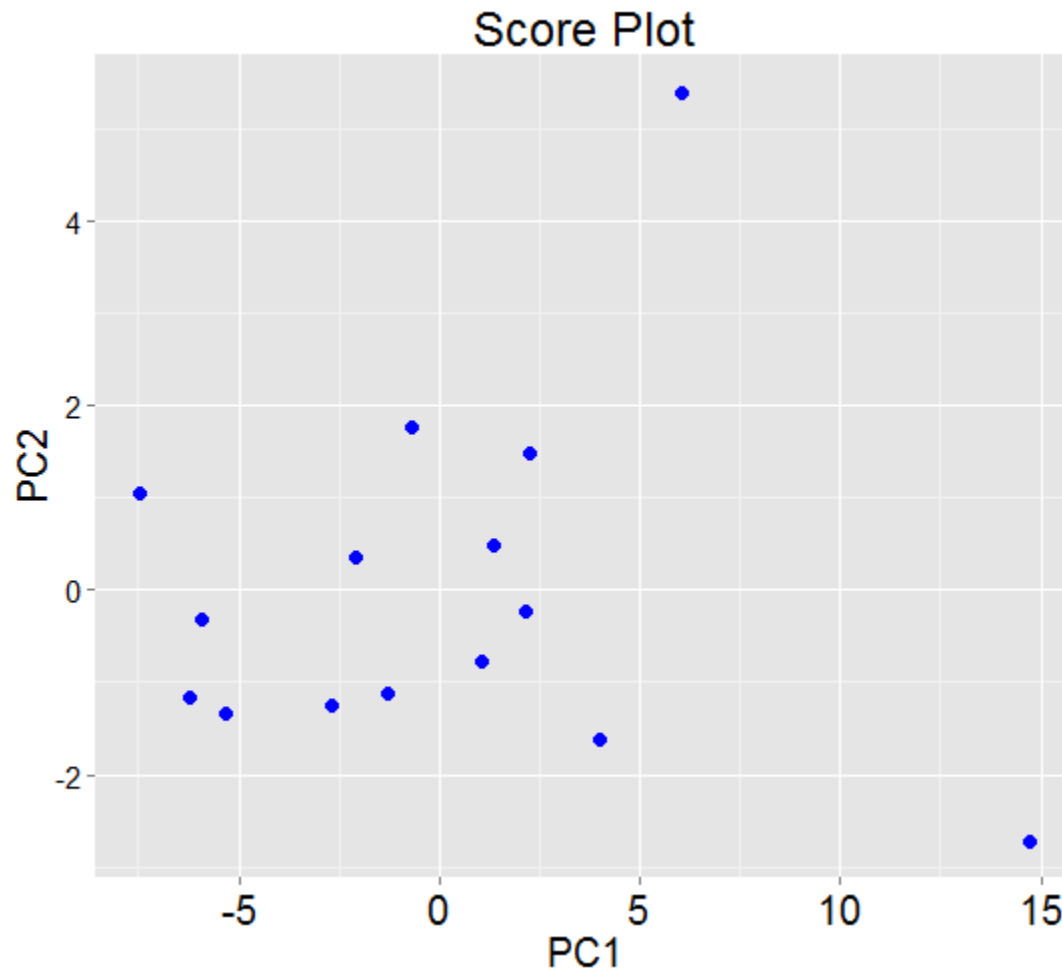
First convert scores to a data frame:

```
pcaBetaScores <- data.frame(pcaBeta$x)
```

Visualize the first two PC's:

```
library(ggplot2)
ggplot(pcaBetaScores, aes(x=PC1, y=PC2)) +
  geom_point(color="blue",
             shape=20,
             size=5) +
  ggtitle("Score Plot") +
  ylab("PC2") +
  xlab("PC1")+
  theme(axis.text.x=element_text(size=20, color="black"),
        axis.text.y=element_text(size=15, color="black"),
        axis.title.y=element_text(size=20),
        title=element_text(size=20))
```


Plot of First Two PC's



The original 44 variables
are projected into 2-
dimensional space

Classification

- For some problems, the subjects have measurements (predictors) and are also classified into 2 or more groups.
- We would like to use the predictors to be able to predict the samples' classification.

Example Data



Animal	Group	t01	t02	t03	t04	t05	t06	...	t42	t43	t44
M01	Control	0.12	0.11	0.11	0.11	0.12	0.12	...	0.09	0.09	0.10
M02	Control	0.10	0.09	0.10	0.10	0.09	0.09	...	0.07	0.07	0.08
M03	Control	0.10	0.10	0.09	0.09	0.09	0.09	...	0.08	0.08	0.07
M04	Control	0.13	0.11	0.12	0.11	0.12	0.12	...	0.12	0.10	0.10
M05	Control	0.10	0.10	0.09	0.10	0.10	0.09	...	0.08	0.07	0.07
M06	Control	0.14	0.13	0.13	0.13	0.12	0.13	...	0.11	0.11	0.11
M07	Control	0.12	0.12	0.12	0.12	0.12	0.12	...	0.11	0.10	0.11
M08	Control	0.11	0.11	0.10	0.10	0.11	0.10	...	0.08	0.09	0.09
M09	Treatment	0.11	0.12	0.10	0.11	0.11	0.10	...	0.09	0.10	0.08
M10	Treatment	0.09	0.09	0.09	0.09	0.08	0.09	...	0.11	0.10	0.11
M11	Treatment	0.04	0.06	0.08	0.07	0.03	0.06	...	0.04	0.03	0.07
M12	Treatment	0.04	0.05	0.10	0.08	0.07	0.09	...	0.10	0.10	0.11
M13	Treatment	0.10	0.10	0.08	0.09	0.09	0.09	...	0.10	0.10	0.09
M14	Treatment	0.09	0.10	0.09	0.09	0.09	0.09	...	0.11	0.08	0.08
M15	Treatment	0.13	0.14	0.12	0.12	0.12	0.11	...	0.11	0.11	0.10

A Few Classification Methods

- Partial least squares
 - Simultaneous dimension reduction and classification
- Recursive partitioning (i.e. Trees)
 - Identifying predictors that optimally partition samples into similar groups
- Random forests
 - An ensemble of recursive partition models

Evaluating Classification Models

- Accuracy
 - Proportion of correctly classified samples
 - Appropriate when there are near equal numbers of samples in each class
- Kappa
 - A comparison of observed agreement versus expected agreement
 - Appropriate when the class sizes are not balanced
- Receiver operating characteristic curve
 - Sensitivity vs. specificity

Partial Least Squares

- Recall: PCA finds linear combinations of the original variables that summarize a *maximum amount of variation* in the original data
 - PCA ignores the response when reducing dimension
- PLS finds linear combinations of the original variables that summarize a maximum amount of *co-variation between the original variables and the categorical response*.
 - PLS uses the response when reducing dimension
 - Like PCA, the predictors need to be centered and scaled

Bring the Data into R & Perform PCA on Centered and Scaled Data

Get data:

```
beta2 <- read.csv("BetaTimecourse2.csv", header=TRUE)
```

Partition Group and variables into separate objects

```
beta2Group <- subset(beta2, select=Group)
dropVars <- names(beta2) %in% c("Animal", "Group")
beta2Vars <- beta2[,!dropVars]
```

Perform PCA on centered and scaled variables

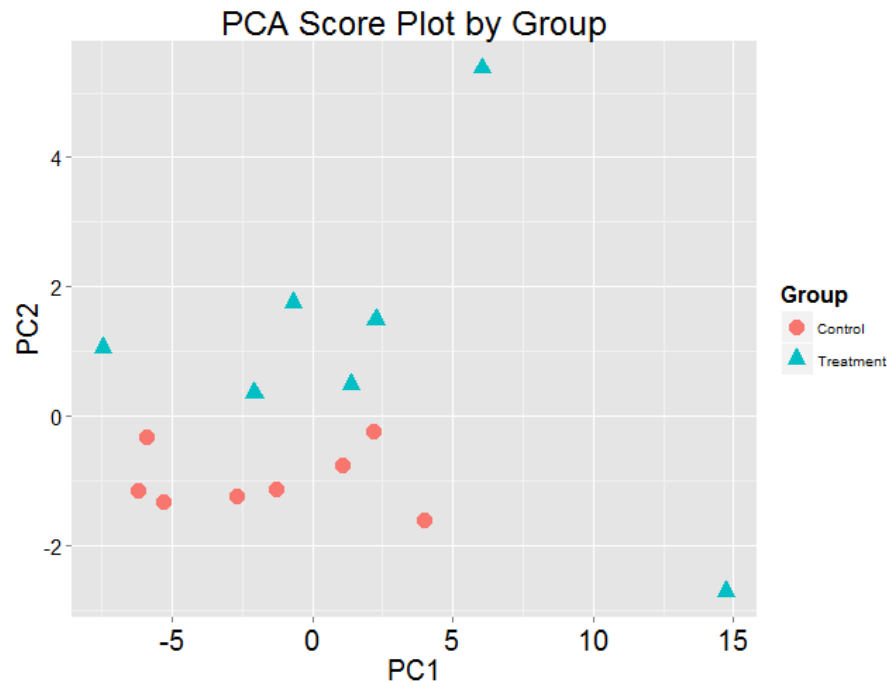
```
pcaBeta2 <- prcomp(x=beta2Vars, center=TRUE, scale.=TRUE,
                  newdata=beta2Vars)
```

Convert to a data frame:

```
pcaBeta2Scores <- data.frame(pcaBeta2$x, Group=beta2Group)
```

Plot of First Two PC's, Colored by Group

```
ggplot(pcaBeta2Scores, aes(x=PC1, y=PC2, color=Group, shape=Group)) +  
  geom_point(aes(color=Group, shape=Group), size=5) +  
  ggtitle("PCA Score Plot by Group") +  
  ylab("PC2") + xlab("PC1") +  
  theme(axis.text.x=element_text(size=20, color="black"),  
        axis.text.y=element_text(size=15, color="black"),  
        axis.title.y=element_text(size=20),  
        title=element_text(size=20))
```



Perform PLS

Get caret package:

```
install.packages("caret")
```

Load package and build PLS model (more on this syntax later):

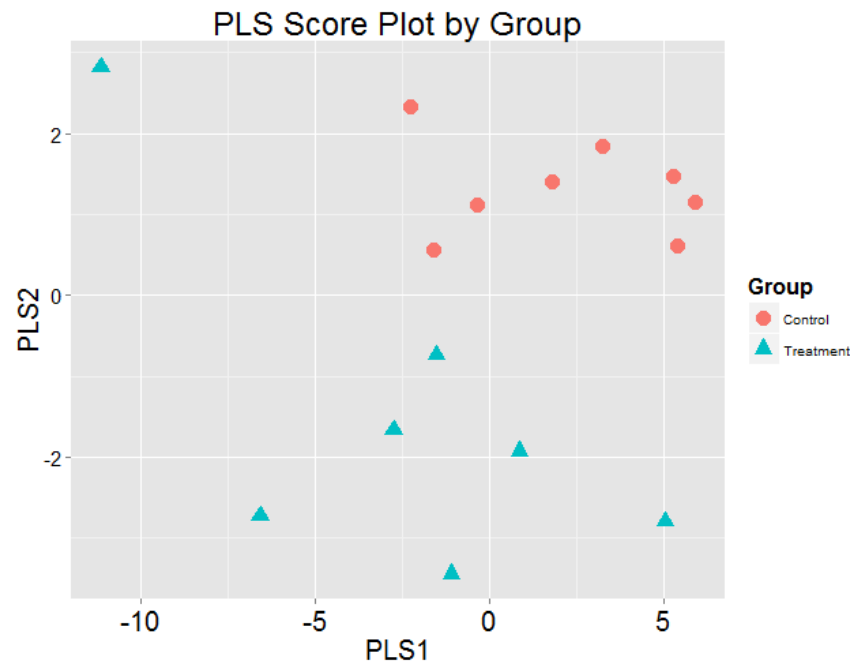
```
library(caret)
set.seed(1002)
plsModel <- train(x=beta2Vars,
                  y=beta2$Group,
                  method="pls",
                  tuneLength=10,
                  preProc = c("center", "scale"),
                  metric = "Accuracy")
```

Convert PLS scores to a data frame:

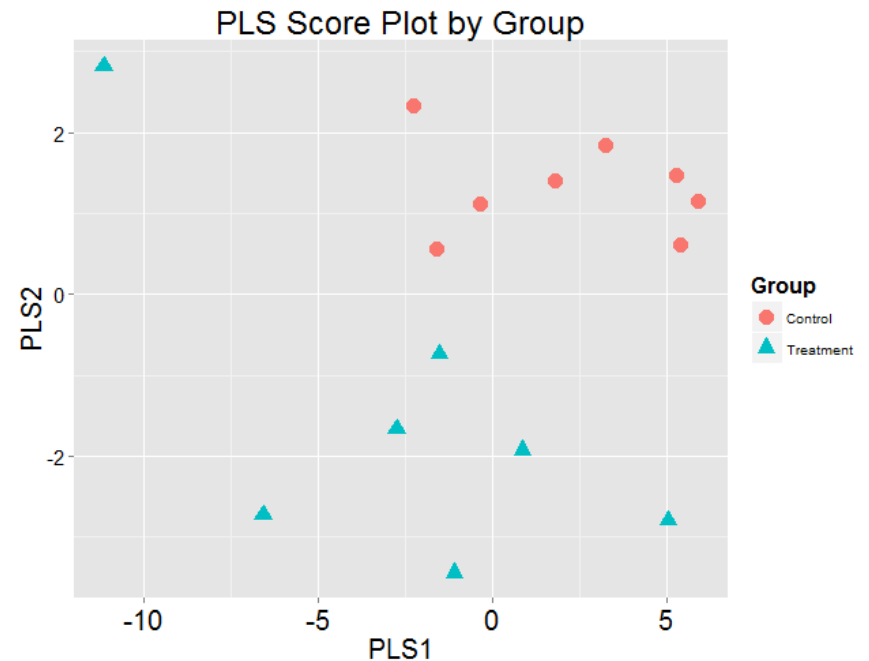
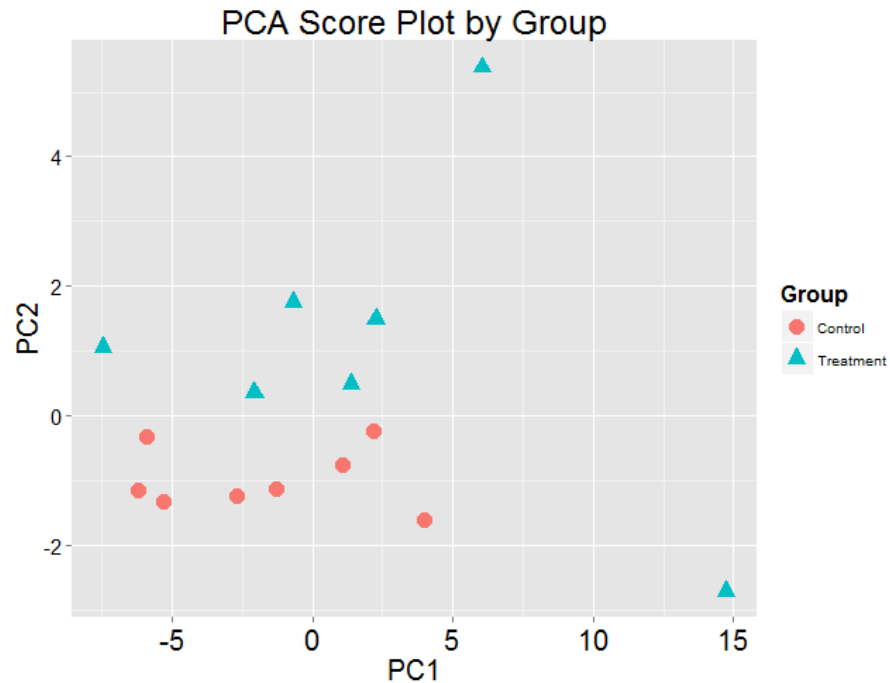
```
plsBeta2Scores <- data.frame(plsModel$finalModel$scores[,1:2],
                             Group=beta2Group)
```

Plot of First Two PC's, Colored by Group

```
ggplot(plsBeta2Scores,  
  aes(x=Comp.1, y=Comp.2, color=Group, shape=Group)) +  
  geom_point(aes(color=Group, shape=Group), size=5) +  
  ggtitle("PLS Score Plot by Group") +  
  ylab("PLS2") + xlab("PLS1") +  
  theme(axis.text.x=element_text(size=20, color="black"),  
        axis.text.y=element_text(size=15, color="black"),  
        axis.title.y=element_text(size=20),  
        title=element_text(size=20))
```



PCA, PLS Comparison



PLS has better *linear* separation between groups. This is a mathematical fact: PLS will always separate groups as well as or better than PCA.

PLS Performance

```
plsModel
```

```
15 samples
44 predictors
 2 classes: 'Control', 'Treatment'
```

```
Pre-processing: centered, scaled
Resampling: Bootstrap (25 reps)
```

```
Summary of sample sizes: 15, 15, 15, 15, 15, 15, ...
```

```
Resampling results across tuning parameters:
```

ncomp	Accuracy	Kappa	Accuracy SD	Kappa SD
1	0.531	0.125	0.232	0.396
2	0.714	0.394	0.187	0.336
3	0.755	0.465	0.158	0.291
4	0.728	0.391	0.167	0.316
5	0.721	0.375	0.179	0.336
6	0.727	0.389	0.181	0.342
7	0.727	0.389	0.181	0.342
8	0.703	0.345	0.187	0.337
9	0.722	0.379	0.183	0.342
10	0.717	0.364	0.185	0.351

```
Accuracy was used to select the optimal model using the largest value.
The final value used for the model was ncomp = 3.
```

The number of PLS components to retain is a “tuning parameter”. We want to optimize the tuning parameter for this data.

Three PLS components produce optimal accuracy for this data

Data for Illustrating Recursive Partitioning and Random Forests

- The previous data set has too few samples for recursive partitioning or random forests.
- R packages can contain data for demonstrations
- Let's get the mdr data set from the caret package

Look to see what data sets are available:

```
data()
```

Load mdr data set:

```
data(mdr)
```

Learn more about the mdr data:

```
?mdr
```

Recursive Partitioning

- Recursive partitioning searches through each predictor to find a value of single predictor that best splits the data into two groups.
 - the best split minimizes misclassification (default)
- For the two resulting groups, the process is repeated until a hierarchical structure (a "tree") is created.
 - trees partition the predictor space into rectangular sections.
- We do not need to center and scale the predictors

Build a Recursive Partitioning Model on mdrd Data

Check balance of categorical response:

```
table(mdrdClass)
```

Build recursive partitioning model:

```
set.seed(1002)
rpartModel <- train(x = mdrdDescr,
                    y = mdrdClass,
                    method = "rpart",
                    tuneLength = 5,
                    metric = "Accuracy")
```

Recursive Partitioning Performance

```
rpartModel
```

```
528 samples
342 predictors
  2 classes: 'Active', 'Inactive'
```

```
No pre-processing
Resampling: Bootstrap (25 reps)
```

```
Summary of sample sizes: 528, 528, 528, 528, 528, 528, ...
```

```
Resampling results across tuning parameters:
```

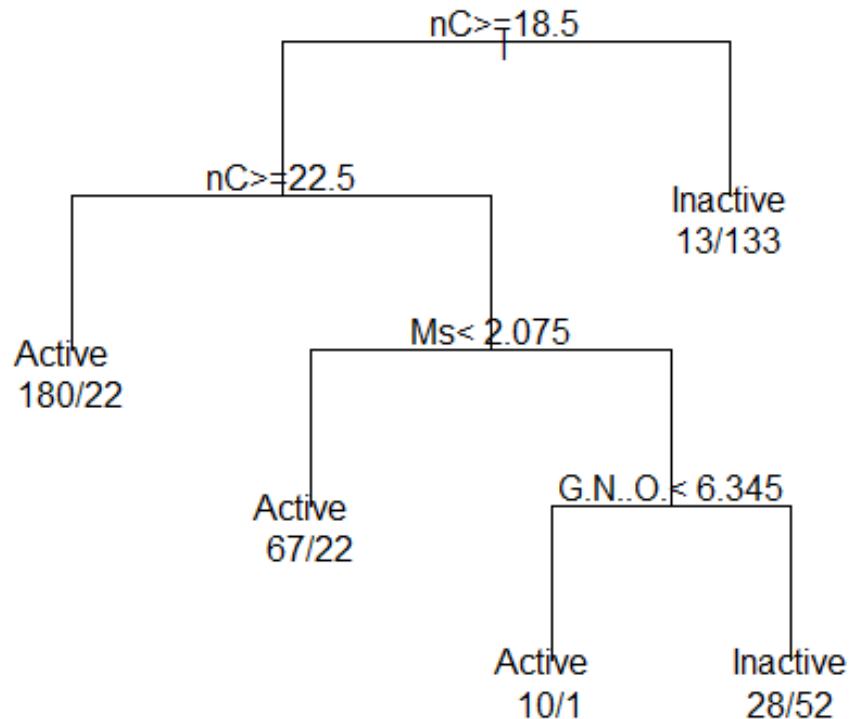
cp	Accuracy	Kappa	Accuracy SD	Kappa SD
0.0087	0.768	0.527	0.0356	0.0694
0.0152	0.771	0.532	0.0392	0.0778
0.0283	0.78	0.548	0.0364	0.0733
0.0326	0.779	0.546	0.0347	0.0697
0.522	0.724	0.413	0.0975	0.218

The complexity of the tree is a “tuning parameter”. We want to optimize the tuning parameter for this data.

```
Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.0283.
```


Visualize the Optimal Model

```
plot(rpartModel$finalModel,  
     compress=TRUE, branch=1, uniform=TRUE, margin=0.1)  
text(rpartModel$finalModel,use.n=TRUE)
```



Random Forests

- A random forest is an ensemble of many recursive partition trees
- The process
 - Do *1000* times (or some large number of times)
 - Create a new version of the original data by randomly selecting samples with replacement (i.e. bootstrapping)
 - Build recursive partitioning model on the new sample using only a randomly selected subset of predictors at each split
 - Aggregate predictions across all 1000 recursive partitioning models
- The number of predictors to use at each split is the tuning parameter

Random Forests Illustration

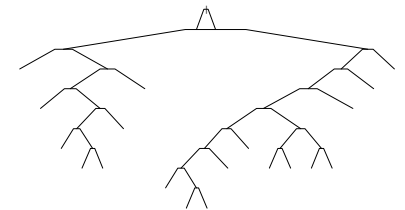
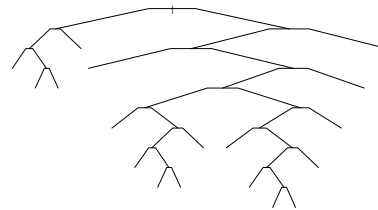
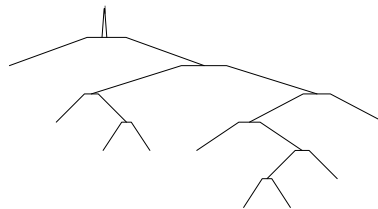
Each data set is a bootstrap sample of the original data

Data set 1

Data set 2

Data set M

Build trees



Predict

Predict

Predict

Final Prediction

Properties of Random Forests

- Variance reduction
 - Averaging predictions across many models provides more stable predictions and model accuracy (Breiman, 1996)
- Robustness to noise
 - All observations have an equal chance to influence each model in the ensemble
 - Hence, outliers have less of an effect on individual models for the overall predicted values

Build a Random Forest Model on mdrr Data

Install the randomForest package:

```
install.packages("randomForest")
```

Build a random forest model:

```
set.seed(1002)
rfModel <- train(x = mdrrDescr,
                 y = mdrrClass,
                 method = "rf",
                 tuneLength = 5,
                 metric = "Accuracy")
```

Random Forest Performance

rfModel

```
528 samples
342 predictors
 2 classes: 'Active', 'Inactive'
```

```
No pre-processing
Resampling: Bootstrap (25 reps)
```

```
summary of sample sizes: 528, 528, 528, 528, 528, 528, ...
```

```
Resampling results across tuning parameters:
```

mtry	Accuracy	Kappa	Accuracy SD	Kappa SD
2	0.802	0.59	0.0306	0.062
87	0.807	0.605	0.0291	0.0593
172	0.805	0.6	0.0296	0.0604
257	0.802	0.594	0.0302	0.0612
342	0.8	0.59	0.0313	0.0629

```
Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 87.
```

The complexity of random forests is mtry. We want to optimize the tuning parameter for this data. Accuracy is better for the random forest model than the recursive partitioning model.

Comments/Questions?

