Stat 437 HW5

Samantha Gregoryk (11559189)

Conceptual exercises: I (support vector machines)

1.1) State the mathematical definition of a hyperplane. Describe the classification rule that is induced by a hyperplane. How does the classification rule involve the normal vector of the hyperplane? (Hint: you can use information on page 12 of Lecture Notes 6 to find the normal vector of a hyperplane and then use information on pages 4 and 5 of Lecture Notes 6.)

- Hyperplane: In the first dimension it is a single point. In the second dimension it is a line, defined by $S = x = (x_1, x_2)^T \epsilon R^2 : ax_1 + bx_2 + d = 0$. In the third dimestion it is a plane, defined by $S = x = (x_1, x_2, x_3)^T \epsilon R^2 : ax_1 + bx_2 + cx_3 + d = 0$ with |a|+|b| > 0.
- Classification rule that is induced by a hyperplane: Since the class label $y_0 \epsilon -1, 1 for x_0 satisfies sgn(y_0) = sgn(< x_0, a > + B_0)$

1.2) Consider a two-class classification problem where observations $\{(x_i, y_i)\}_{i=1}^n$ can be completely separated by a hyperplane. Consider a hyperplane $S = \{x \in \mathbb{R}^p : \langle x, \alpha \rangle + \beta_0 = 0\}$ with direction α and intercept β_0. Explain why the distance from x_i to S is

$$\text{dist}(x_i, S) = y_i(\langle x_i, \alpha \rangle + \beta_0)$$

when $\|\alpha\| = 1$? (Hint: you can read through pages 11 and 12 of Lecture Notes 6 and watch corresponding lecture video clips.)

- The distance from x_i to S is

  $$\text{dist}(x_i, S) = y_i(\langle x_i, \alpha \rangle + \beta_0)$$

  when $\|\alpha\| = 1$ because for any point $x_i \epsilon R^P$ with the class label $y_0 \epsilon -1, 1$ for x_i satisfying $sgn(y_0) = sgn((x_i, a) + B_0)$, when $||a|| = 1$, we get $dist(x, S) = y_0((x_i, a) + B_0)$

1.3) Consider a two-class classification problem where observations $\{(x_i, y_i)\}_{i=1}^n$ can be completely separated by a hyperplane. Consider a hyperplane $S = \{x \in \mathbb{R}^p : \langle x, \alpha \rangle + \beta_0 = 0\}$. Why there are infinitely many separating hyperplanes for these observations? What is the optimization problem that the maximal margin classifier tries to solve? State the optimization problem mathematically and explain the meaning of each term in the mathematical formulation. (Hint: you need to first set up notations and then you can use information on page 13 of Lecture Notes 6.) Why does the optimization problem have the constraint $\|\alpha\| = 1$? (Hint: you can use partial answer to 1.2) above.) Explain why the optimal hyperplane of the maximal margin classifier is equal distance from either class of observations. (Hint: you can use information on pages 9 and 10 of Lecture Notes 6.)

- There are infinitely many separating hyperplanes for these observations because data is linearly separable and observations are perfectly separately by a hyperplane.
- The optimization problem that the maximal margin classifier tries to solve is to choose the hyperplane farthest away from the training observations for minimum distance of an observation in the set to the hyperplane (margin).
- Optimization problem mathematically: $min(y_i((x_i, a) + B_0)) \geq M$
- The meaning of each term: : distnace of training set T to $S(a, B_0)$ is $min(y_i((x_i, a) + B_0))$ and $y_i((x_i, a) + B_0)$ is the distance of x_i to $S(a, B_0)$.
- The optimization problem have the constraint $\|\alpha\| = 1$ because it ensures that the distance of x_i to $S(a, B_0)$ is exactly $y_i((X_i, a) + B_0)$.

1.4) Consider a two-class classification problem where observations can be completely separated by a hyperplane. What are support vectors of the maximal margin classifier? Explain how you move support vectors to change and not to change the maximal margin classifier, respectively.

- Support vectors of the maximal margin classifier are the generalization of the maximal margin classifer to the non-seperable case.
- You move support vectors to change or not change the maximal margin classifier by moving the vectors since the classifiers depends on the vectors more than the observations in the training set.

1.5) Consider a two-class classification problem where observations $\{(x_i, y_i)\}_{i=1}^n$ can not be completely separated by a hyperplane. What optimization problem does a support vector classifier (SVC) try to solve? State it mathematically and explain the meaning of each term in the mathematical formulation. Explain how the value of a slack variable reveals how its associated observation is classified by the resulting SVC, and explain how the value of the tolerance affects classification of x_i's, the number of support vectors, and the margin of the resulting SVC. (Note: please do NOT just copy contents from the lecture notes and paste them as your answers.)

- The optimization problem a support vector classifier (SVC) trys to solve is developing a hyperplane that almost separates the classes using a soft-margin.

- State mathematically: $\{1 \leq i \leq n, y_i((X_i, a) + B_0) \geq M(1 - \epsilon_i)$ $1 \leq i \leq n, \epsilon_i \geq 0 : \sum \epsilon_i \leq C$ for some $C > 0$
- The meaning of each term in the mathematical formulation: distance from x_i to $S(a, B_0)$: $y_i((X_i, a) + B_0)$. Proportional margin: $M(1 - \epsilon_i)$. Amount of violation: $1 \leq i \leq n, \epsilon_i \geq 0 : \sum \epsilon_i \leq C$.
- The value of a slack variable reveals how its associated observation is classified by the resulting SVC by allowing individual observations to be on the wrong side of the margin or hyperplane (0, $\epsilon(0, 1]$, >1).
- The value of the tolerance affects classification of x_i's, the number of support vectors, and the margin of the resulting SVC by becoming more stable to changes in individual observations but have larger classification error.

1.6) Consider a two-class classification problem where observations $\{(x_i, y_i)\}_{i=1}^n$ can not be completely separated by a hyperplane. When constructing an SVC by solving the optimization problem via Lagrange multipliers, there is a "cost" parameter C. Explain how the value of the cost C affects classification of x_i's, the number of support vectors, and the margin of the resulting SVC. Is this C the same as the tolerance mentioned in 1.5)?

- The value of the cost C affects classification of x_i's, the number of support vectors, and the margin of the resulting SVC by having a wider margin with smaller C and C $= \infty$ for the seperable case.
- This C the same as the tolerance mentioned in 1.5.

1.7) Consider a two-class classification problem where training observations $\{(x_i, y_i)\}_{i=1}^n$ can not be completely separated by a hyperplane. When the decision boundary between the two classes is nonlinear, what can you to to an SVC in order to deal with this situation, and what are some disadvantages of what you propose to do? Is it true that an SVM is able to deal with this situation and that it does so by implicitly enlarging the feature space using a kernel that can be different from the Euclidean inner product? Provide a linear representation of an SVM, and comment on how this representation is different from and similar to that for an SVC, respectively.

- When the decision boundary between the two classes is nonlinear, you can enlarge the original feature by including nonlinearly transformed feature variables or apply a linear classifier (SVC) via Euclidean inner product to an SVC in order to deal with this situation.
- Some disadvantages are it may lead to unmanageable computations in the optimization problem.
- It is true that an SVM is able to deal with this situation and that it does so by implicitly enlarging the feature space using a kernel that can be different from the Euclidean inner product.
- Linear representation of an SVM: $f(x; B, B_0) = (x, B) + B_0$ with $B = (B_1, ..., B_p)^T$.

1.8) Describe how to conduct multi-class classification using SVMs.

- Conduct multi-class classification using SVMs: Use one-versus-One classification approach by constructing $(s$
  2) SVMs and assign x_0 to the class. Use one-versus-all by constructing s SVMs, each of which compares one of the s classes with the remaining s-1 classes.

Conceptual exercises: II (neural networks)

2.1) Describe how derived features are obtained by a vanilla, feedforward neural network that has 3 layers in total and 1 hidden layer.

- Derived features are obtained by a vanilla, feedforward neural network that has 3 layers in total and one hidden layer by having 1 input layer, 1 hidden layer, and 1 input layer.

2.2) Provide a criterion that is used to train a neural network for classification and for regression, respectively.

- Criterion that is used to train a neural network for classification is the cross-entropy $R(\theta) = -\sum\sum(y_(ik) log f_k(x_i))$.

- Criterion that is used to train a neural network for regression is the sum of squared errors $R(\theta) = -\sum\sum(y_(ik) - f_k(x_i))^2$.

2.3) What are some issues on training a neural network by optimizing a criterion you presented in 2.2), and how to deal with them?

- Some issues on training a neural network by optimizing a criterion are first, there are many things we do not understand about NNs in general. A NN is often over-parametrized and we do not seek for a global minimizer for overfitting. Regularization on weights is recommended to mitigate overfitting or a validation dataset is used to check for overfitting. Also a minimizer may depend on the initial values for the weights. Trying a number of random starting configurations for the NN is recommended.

Conceptual exercises: III (principal component analysis)

Assume there are p feature variables $X_1, \ldots, X_p$ that are stored in the vector $X = (X_1, \ldots, X_p)^T$. Let $\mathbf{X}$ be an $n \times p$ data matrix whose ith row is the ith observation on X. Assume the covariance matrix of X is Σ.

3.1) Describe in detail the population version of principal component analysis (PCA).

- Population version of PCA uses probabilistic formulation p stochastically linearly independent feature variables $X_1, ..., X_p$ and feature vector $X = (X_1, ..., X_p)^T$ for some $a_1, ..., a_p \epsilon R$ implies that $a_i = 0$ for all i=1,. . . ,p with probability 1.

3.2) Provide the sample covariance matrix of X that is obtained from $\mathbf{X}$. Describe in detail the data version of PCA.

- Sample covariance matrix of X that is obtained from $\mathbf{X}$: $S = X^T X \epsilon R^{(PXP)}$.
- Data version of PCA employs SVD of the data matrix and does not employ a probabilistic model and has two equivalent formulations.

3.3) In the population version of PCA, the first principal component is a scalar random variable, whereas in the data version of PCA, we have n scores for the first principal component. How are the first principal component and its n scores related?

3.4) What does a biplot plot? How can you discover patterns in data using a biplot?

- A biplot plots by using 2 coordinate systems that share the same origin. One coordinate system has x-coordinate for scores of PC i and y-coordinate for scores of another PC j and has the orientation and layout of the conventional x-y coordinate system. The other coordinate system has x-coordinate for loadings of PC i and y-coordinate for loadings of another PC j, where the x-axis tick marks are on the top and y-axis tick marks are on the right.
- You can discover patterns in data using a biplot by

3.5) When implementing the data version of PCA based on $\mathbf{X}$, is it recommended to center and scale the observations in $\mathbf{X}$? If so, how and why?

- It is recommended to center and scale the observations in $\mathbf{X}$ by using the scale function with center equaling true because total variability of only centered data depends on magnitudes of measurements on features.

3.6) What is a criterion to use to choose the number of principal components?

- A criterion to use to choose the number of PCs is to choose a minimum number of PCs that together explain a given cumlative proportion of variance.

3.7) Consider the scalar random variable $w = a^T X$ for $a \in \mathbb{R}^p$. We want to find $a \in \mathbb{R}^p$ for which the variance of w is maximized. Explain why a should be an eigenvector associated with the largest eigenvalue λ_1 of Σ.

- a should be an eigenvector associated with the largest eigenvalue λ_1 of Σ because

3.8) State the model and optimization problem PCA tries to solve when it is interpreted as the best linear approximate to $\mathbf{X}$ under the Frobenius norm among all subspace of dimension $q < p$. How is this optimization problem related to regression modeling based on the least squares method?

- The model and optimization problem PCA tries to solve when it is interpreted as the best linear approximate to $\mathbf{X}$ under the Frobenius norm among all subspace of dimension $q < p$ is to propose and conduct nonlinear dimension reduction if we do not restrict ourselves to use linear subspaces.

Applied exercises

Consider the data set `iris` from the R library `ggplot2`. Here is the instructor's ggplot2 version

```
packageVersion("ggplot2")
```

```
## [1] '3.3.2'
```

You can use `help(iris)` to obtain some help information on this data set, or you can do the following:

```
library(ggplot2)
library(e1071)
library(neuralnet)
library(ggfortify)

data(iris)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
unique(iris$Species)
```

```
## [1] setosa     versicolor virginica
## Levels: setosa versicolor virginica
```

From the `iris` data set, pick all observations for the subspecies `setosa` or `versicolor`. This gives a subset of 100 observations. From this subset, use `set.seed(123)` to randomly select 40 observations for each of the 2 subspecies, and put the 80 observations thus obtained into a training set. The remaining 20 observations in the subset then form a test set.

```
set.seed(123)

trainId = c(sample(1:50, 40), sample(51:100, 40))
testId = (1:100)[-trainId]
trainingSet = iris[trainId, ]
testSet = iris[testId, 1:5]
testLabs = iris$Species[testId]
trainingLabs = iris$Species[trainId]
```

(4.1) Build an SVM using the training set with cost $C = 0.1$ and apply the obtained model to the test set. Report classification results on the test set and provide needed visualizations. (Note: `plot` for `svm` is not designed for more than 2 features, i.e., when an `svm` is built using more than 2 features and you apply `plot` to an `svm` object, you will get an error.)

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.0.5
```

```
svmFit <- svm(Species ~ ., data = trainingSet, cost = 0.1)
summary(svmFit)
```

```
##
## Call:
## svm(formula = Species ~ ., data = trainingSet, cost = 0.1)
##
##
## Parameters:
```

```
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.1
##
## Number of Support Vectors:  40
##
##  ( 20 20 )
##
##
## Number of Classes:  2
##
## Levels:
##  setosa versicolor virginica
```

```
testProb <- predict(svmFit, testSet)
table(predict = testProb, truth = testSet$Species)
```

```
##             truth
## predict      setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0         10         0
##   virginica       0          0         0
```

(4.2) Build an SVM using the training set by 10-fold cross-validation and by setting `set.seed(123)`, in order to find the optimal value for the cost C from the range:

```
set.seed(123)

tune.out <- tune(svm, Species ~ ., data = trainingSet, ranges = list(cost = c(0.001,
    0.01, 0.1, 1, 5, 10, 100)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0
##
## - Detailed performance results:
##    cost error dispersion
## 1 1e-03 0.275  0.3574602
## 2 1e-02 0.275  0.3574602
## 3 1e-01 0.000  0.0000000
## 4 1e+00 0.000  0.0000000
## 5 5e+00 0.000  0.0000000
## 6 1e+01 0.000  0.0000000
## 7 1e+02 0.000  0.0000000
```

```
predCV <- predict(tune.out$best.model, trainingSet, ranges = list(cost = c(0.001,
    0.01, 0.1, 1, 5, 10, 100)))
table(predicted = predCV, truth = trainingSet$Species)
```

```
##               truth
## predicted     setosa versicolor virginica
##   setosa          40          0         0
##   versicolor       0         40         0
##   virginica        0          0         0
```

Apply the model to the test set, and report classification results on the test set. Do you think an SVM with a nonlinear decision boundary should be used for this classification task? If so, please use an SVM with a radial kernel whose parameters are determined by 10-fold cross-validation on the training set and by setting `set.seed(123)`.

```
predCV <- predict(tune.out$best.model, trainingSet)
table(predicted = predCV, truth = trainingSet$Species)
```

```
##               truth
## predicted     setosa versicolor virginica
##   setosa          40          0         0
##   versicolor       0         40         0
##   virginica        0          0         0
```

(4.3) Use the training set, use `set.seed(123)`, and apply 5-fold cross-validation to build an optimal neural network model with 2 hidden layers of 5 and 7 hidden neurons, respectively. Apply the optimal neural network model to the test set and report classification results. Note that you need to make sure you know how the class labels are ordered by R and that this is explained in the lecture video "Stat 437 Video 27b: neural network example 1".

```
set.seed(123)

m = 5
folds = sample(1:m, nrow(trainingSet), replace = TRUE)

nnTrain = neuralnet(Species ~ ., trainingSet, hidden = c(5, 7),
    act.fct = "logistic", linear.output = FALSE)
```

```
## Error in neuralnet(Species ~ ., trainingSet, hidden = c(5, 7), act.fct = "logistic", : coul
```

```r
plot(nnTrain, show.weights = F, information = F, intercept = F,
    rep = "best", col.hidden = "blue")
```

```
## Error in plot(nnTrain, show.weights = F, information = F, intercept = F, : object 'nnTrain'
```

```r
nnModels = vector("list", m)
testErrorCV = double(m)
for (s in 1:m) {
    trainingTmp = trainingSet[folds != s, ]
    testTmp = trainingSet[folds == s, ]
    testLabsTmp = trainingLabs[folds == s]
    nnetTmp = neuralnet(Species ~ ., trainingTmp, hidden = c(5,
        7), act.fct = "logistic", linear.output = FALSE)
    nnModels[[s]] = nnetTmp
    ypred = neuralnet::compute(nnetTmp, testTmp)
    yhat = ypred$net.result
    # assign labels
    SpeciesEst = data.frame(labelEst = ifelse(max.col(yhat[,
        1:2]) == 1, "setosa", ifelse(max.col(yhat[, 1:2]) ==
        2, "versicolor")))
    SpeciesEst = factor(SpeciesEst[, 1])
    nOfMissObs = sum(1 - as.numeric(testLabsTmp == SpeciesEst))
    terror = nOfMissObs/length(testLabsTmp)
    testErrorCV[s] = terror
}
```

```
## Error in neuralnet(Species ~ ., trainingTmp, hidden = c(5, 7), act.fct = "logistic", : coul
```

```r
optNNnumber = min(which(testErrorCV == min(testErrorCV)))
optNNnumber
```

```
## [1] 1
```

```r
optNNModel = nnModels[[optNNnumber]]

NNOptPred = neuralnet::compute(nnModels[[optNNnumber]], testSet)
```

```
## Error in cbind(1, pred) %*% weights[[num_hidden_layers + 1]]: requires numeric/complex matri
```

```r
yhatOPt = NNOptPred$net.result
```

```
## Error in eval(expr, envir, enclos): object 'NNOptPred' not found
```

```
SpeciesEstOpt = data.frame(labelEst = ifelse(max.col(yhatOPt[,
    1:2]) == 1, "setosa", ifelse(max.col(yhatOPt[, 1:2]) == 2,
    "versicolor")))
```

```
## Error in as.matrix(m): object 'yhatOPt' not found
```

```
SpeciesEstOpt = factor(SpeciesEstOpt[, 1])
```

```
## Error in factor(SpeciesEstOpt[, 1]): object 'SpeciesEstOpt' not found
```

```
nOfMissObsOPt = sum(1 - as.numeric(testLabs == SpeciesEstOpt))
```

```
## Error in eval(expr, envir, enclos): object 'SpeciesEstOpt' not found
```

```
terrorOPt = nOfMissObsOPt/length(testLabs)
```

```
## Error in eval(expr, envir, enclos): object 'nOfMissObsOPt' not found
```

```
table(SpeciesEstOpt, testLabs)
```

```
## Error in table(SpeciesEstOpt, testLabs): object 'SpeciesEstOpt' not found
```

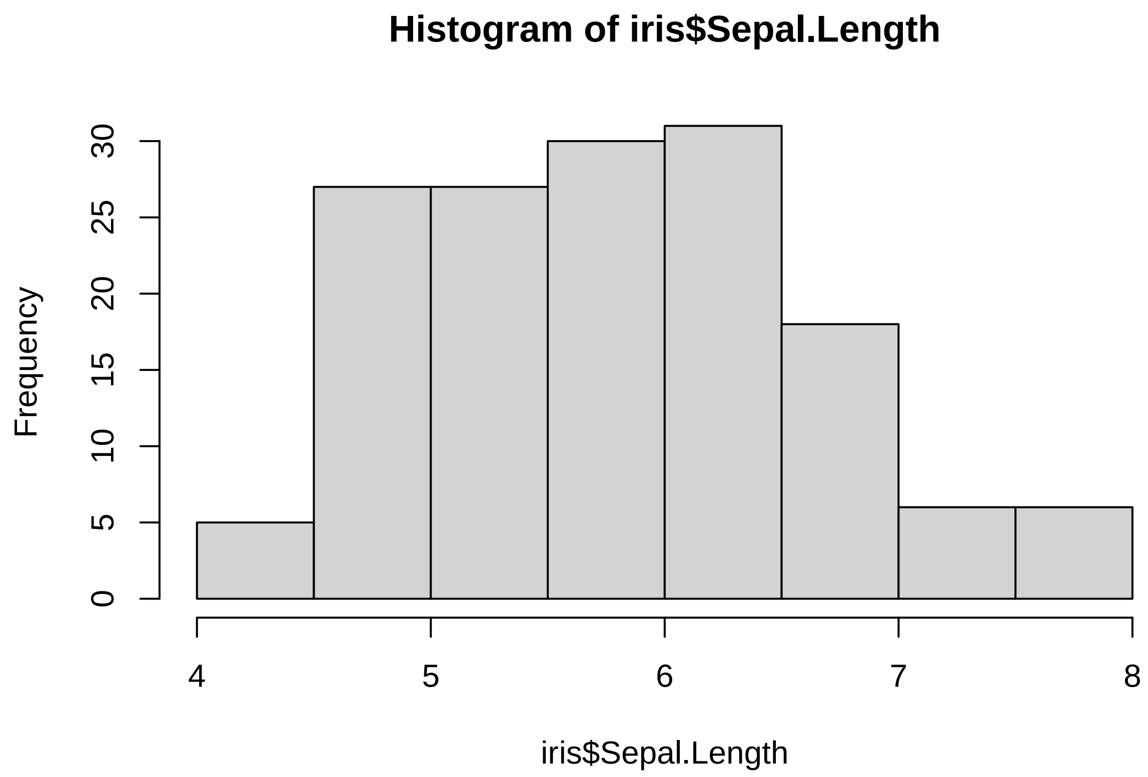
(4.4) Apply PCA to all features of the **full data set** `iris`. Plot the first two principal components against each other by coloring each point on the plot by its corresponding subspecies. Plot the cumulative percent of variation explained by all (successively ordered) principal components.

```
library(ggfortify)
```

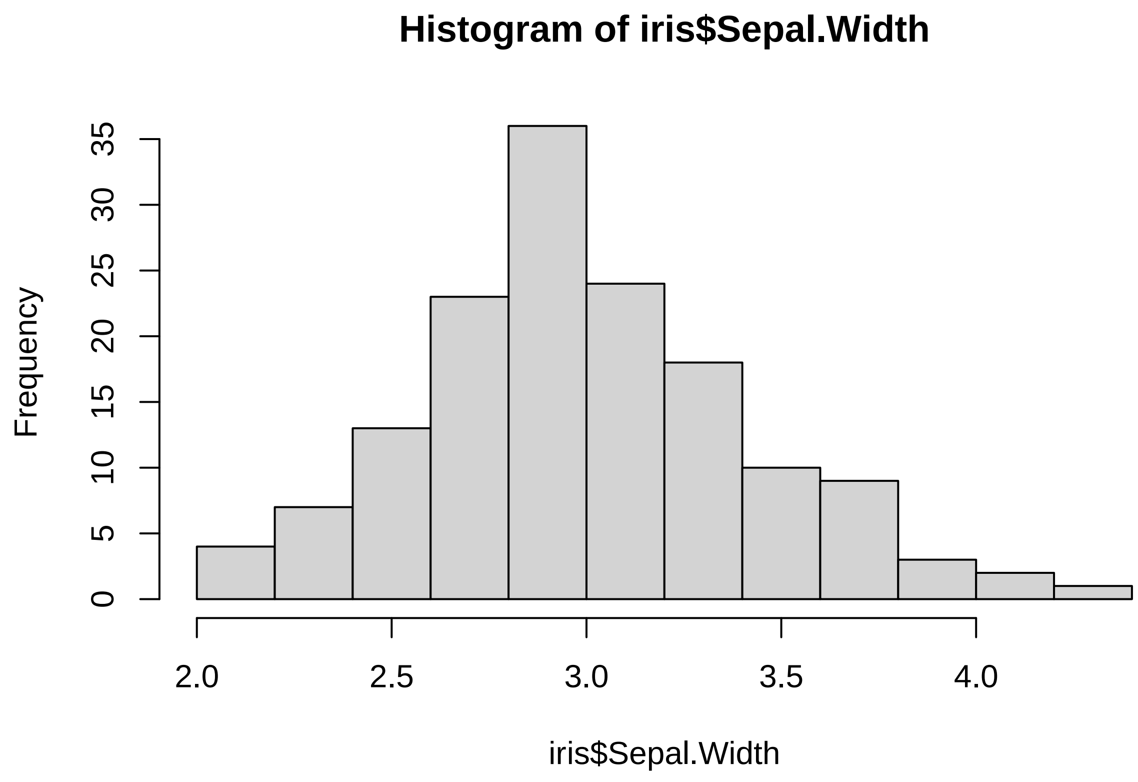
```
## Warning: package 'ggfortify' was built under R version 4.0.5
```

```
## Loading required package: ggplot2
```

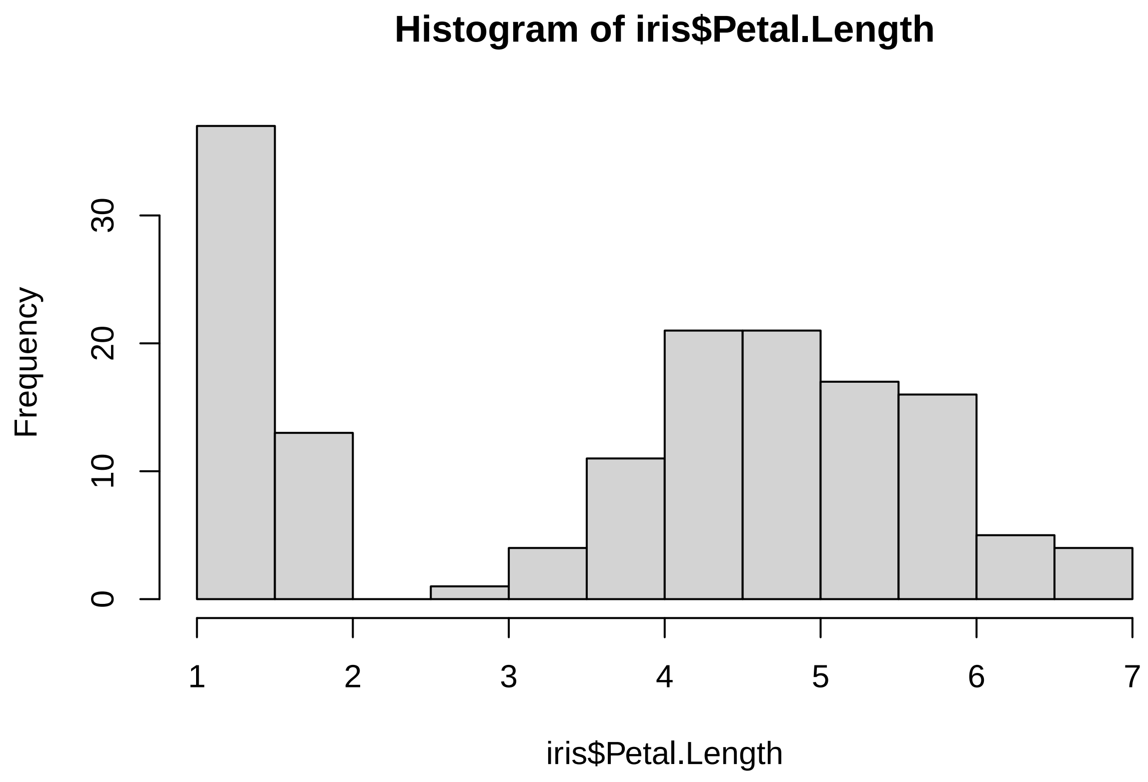
```
hist(iris$Sepal.Length)
```



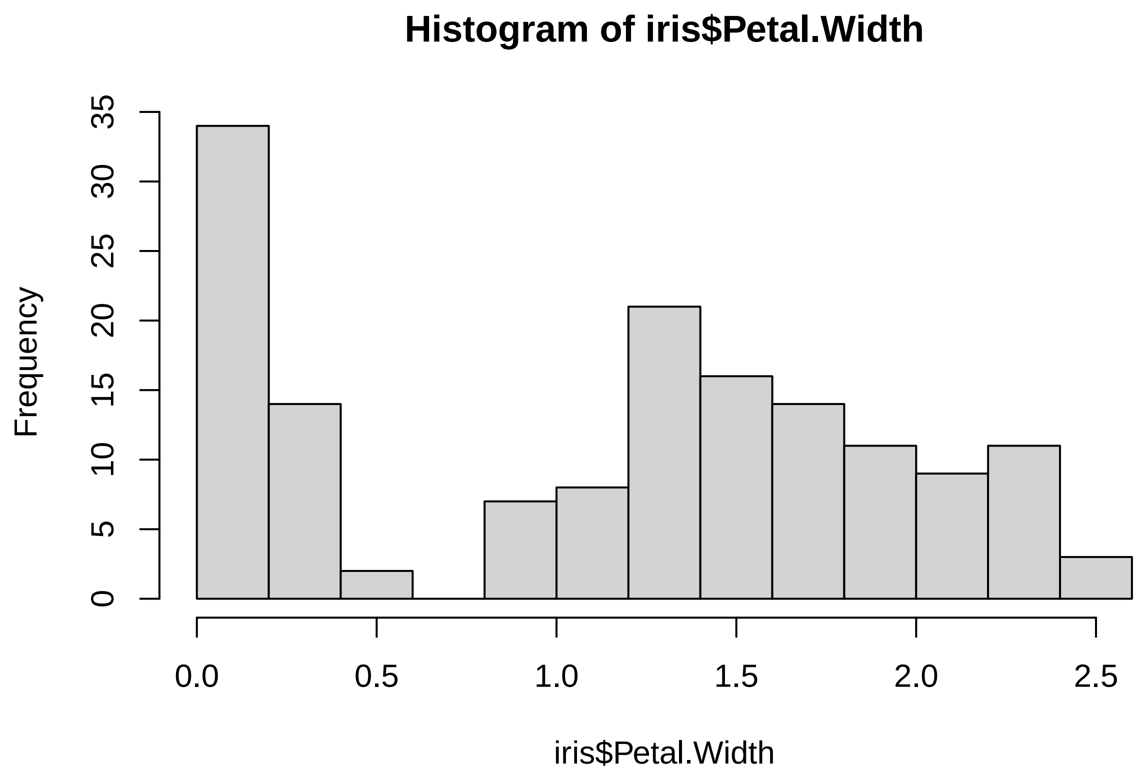
```
hist(iris$Sepal.Width)
```



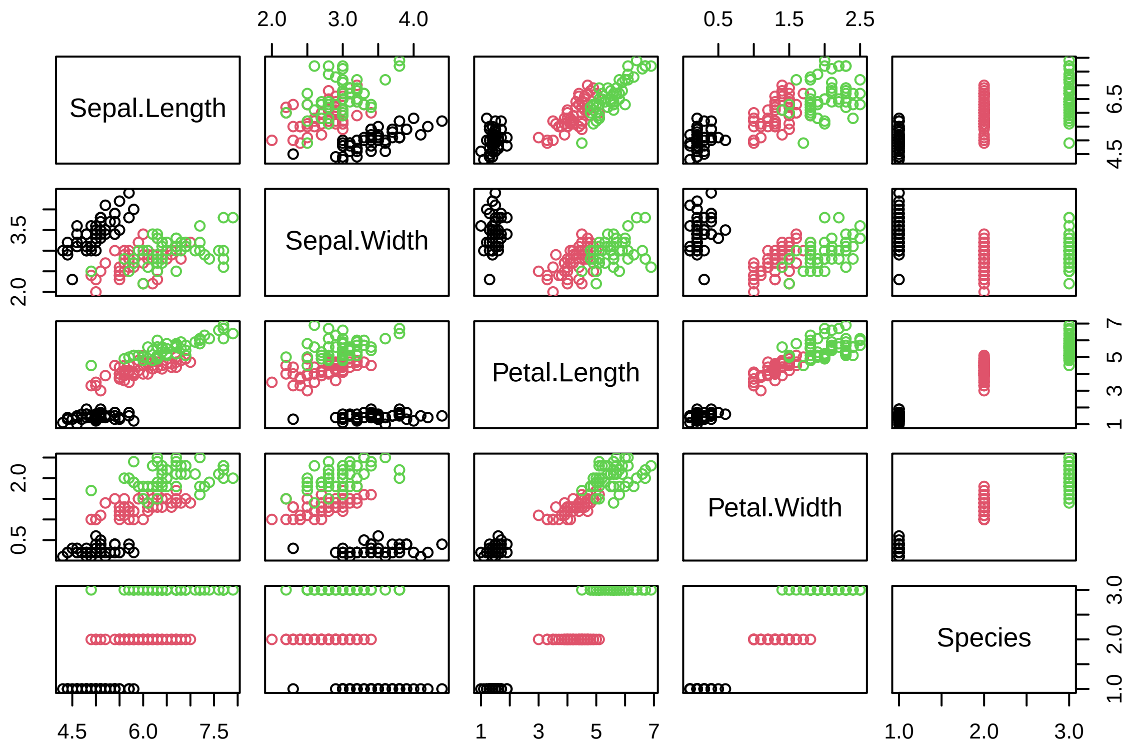
```
hist(iris$Petal.Length)
```



```
hist(iris$Petal.Width)
```



```
pairs(iris, col = iris$Species)
```



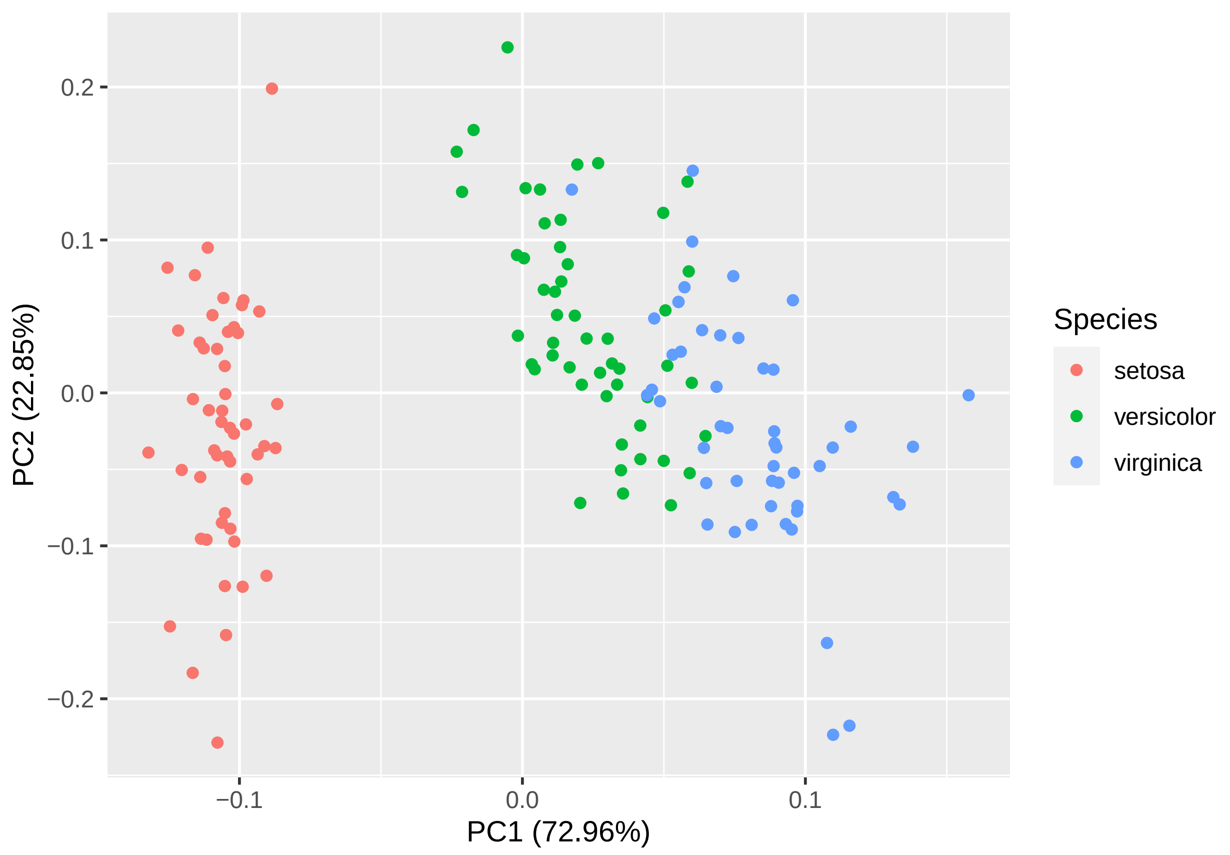
```
cor(iris[, 1:4])
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
## Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
## Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
## Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
```

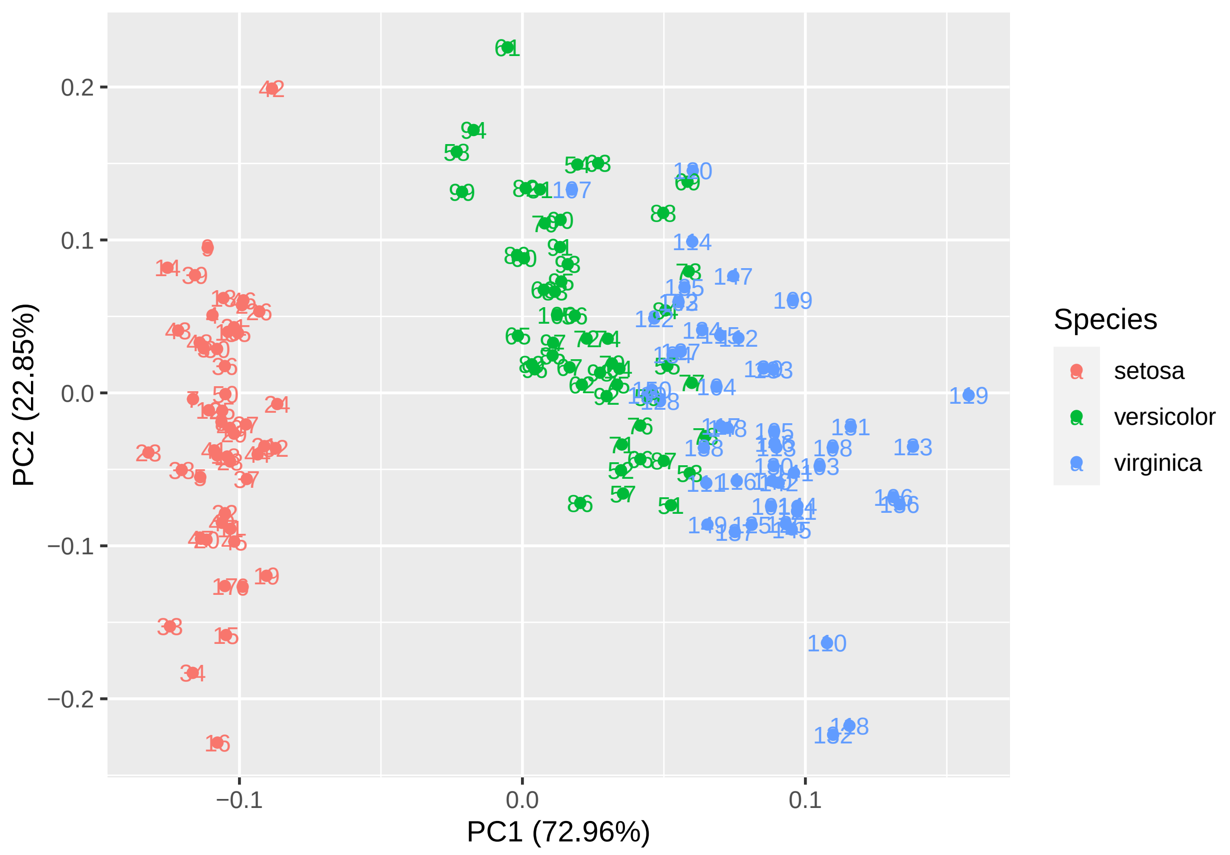
```
df <- iris[1:4]
pca <- prcomp(df, scale. = TRUE)

autoplot(pca, data = iris, colour = "Species")
```

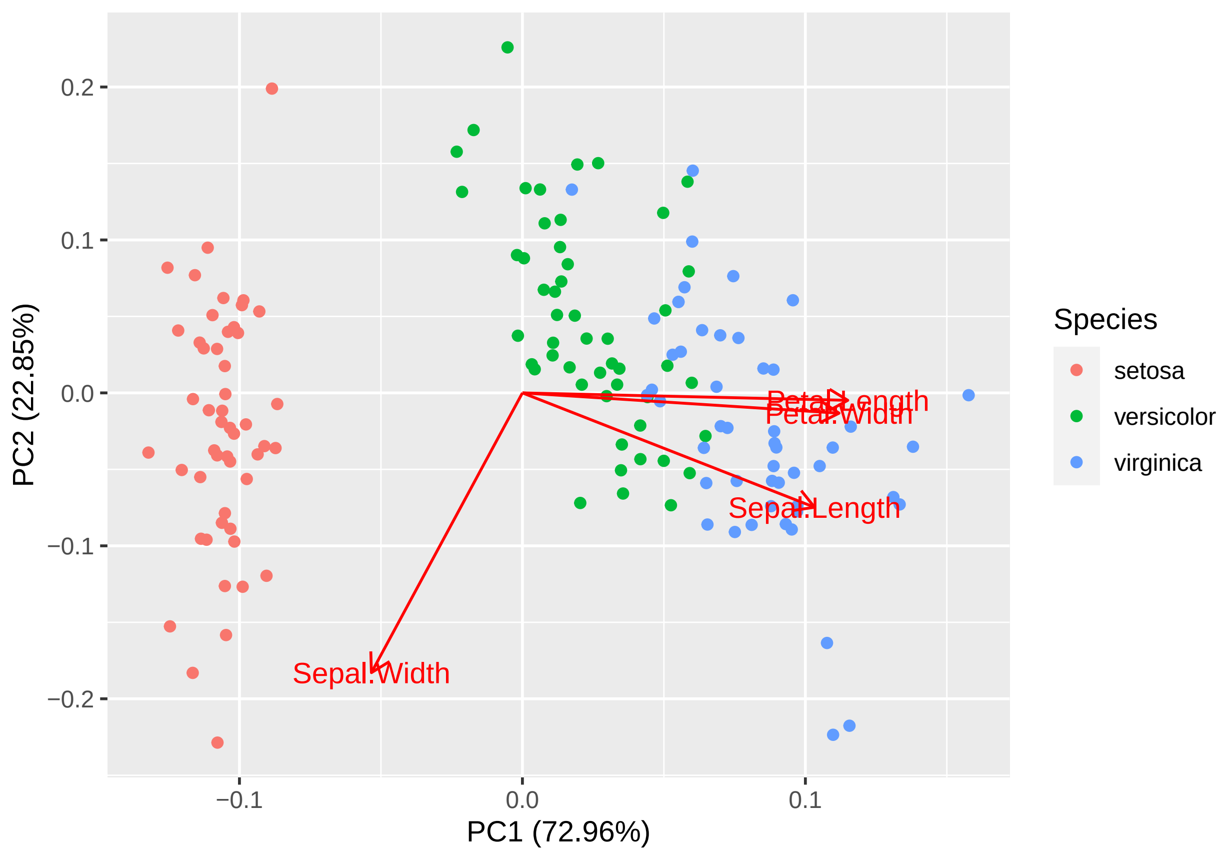
```
## Warning: `select_()` is deprecated as of dplyr 0.7.0.
## Please use `select()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```



```
autoplot(pca, data = iris, colour = "Species", label = TRUE,
    label.size = 3)
```

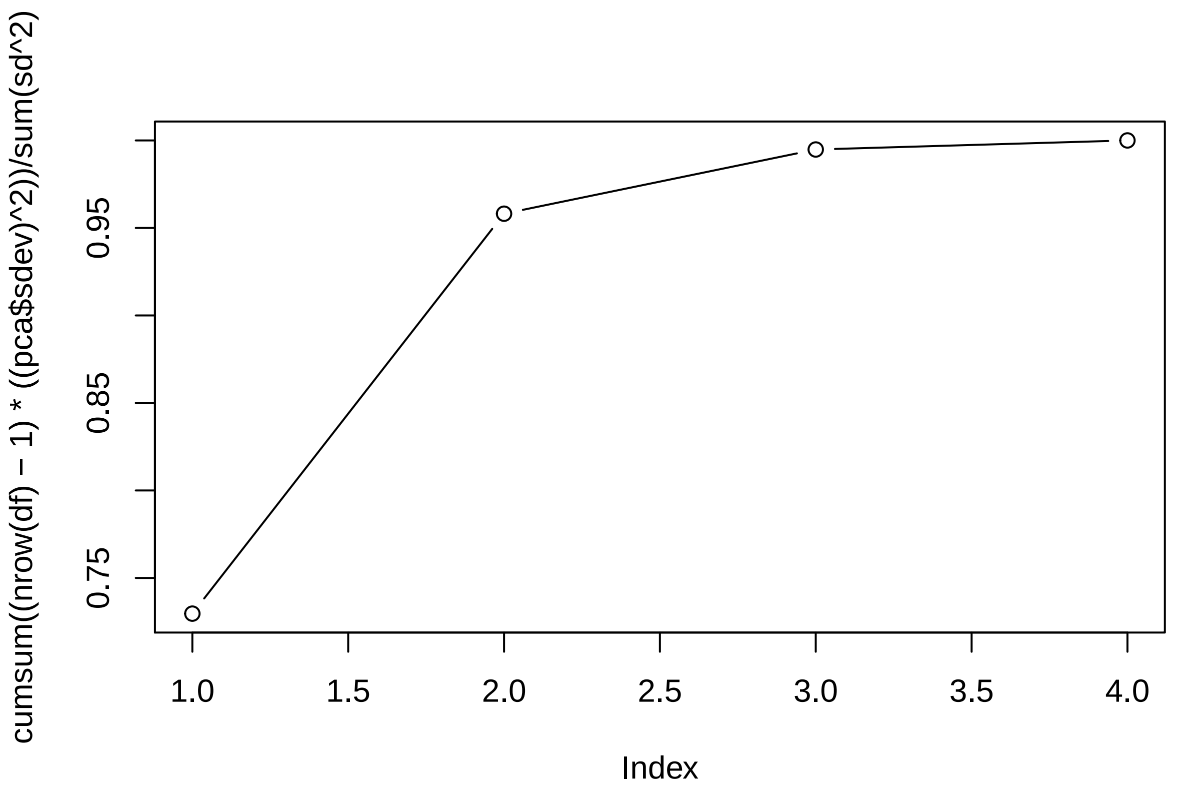


```r
autoplot(pca, data = iris, colour = "Species", loadings = TRUE,
    loadings.label = TRUE)
```



```
sd = scale(df, center = TRUE, scale = TRUE)

plot(cumsum((nrow(df) - 1) * ((pca$sdev)^2))/sum(sd^2), type = "b")
```



Do these principal components reveal any systematic pattern on the features for any subspecies?

- The first loading vector places approximately equal weight on Petal Width and Length, partially on Sepal Length and less weight on Sepal Width.