Using the BioSeqClass Package

Hong Li[‡][*]

April 14, 2011

[‡]Key Lab of Systems Biology
Shanghai Institutes for Biological Sciences
Chinese Academy of Sciences, P. R. China

Contents

1 Overview

There are now 863 completely sequenced genomes of cellular organisms in NCBI genome database. Nevertheless, functional annotation drops far behind sequencing because functional valida-tion experiments are time-consuming and costly. Taken model organism Homo

[*] sysptm@gmail.com

sapiens, Mus musculus and Saccharomyces cere-visiae as examples, only 16and 18annotations in Gene Ontology), respectively. Thus computational methods for predicting function is still a fun-damental complement. The most common com-putation approach is biological sequence based classification, since sequence information is still the most abundant and reliable. Se-quence based classification has been used in: discovering new microRNA candidates, predicting transcription factor binding sites , detecting protein post-translational modification sites , and so on.

Features and models are two basic factors for classification. Features generally are numerical values that can be used to distinguish different classes. Therefore it is preferable to select features that can achieve better and faster classification. Classification models are built from features by various algorithms, and it is necessary to evaluate its prediction ability by cross validation or jackknife test. For biological sequences, there are additional steps: one is to reduce homolog sequences which might result in overestimation of prediction accuracy, and then another most important step is to convert sequences into numerical features. Thus, the general workflow for sequence-based classifications includes (Figure 1): reduce homolog sequences; extract features from sequences and code them to numerical values; evaluate and select features; build classification model and evaluate its performance.

Here we present an R package (BioSeqClass) to carry out the general workflow for biological sequence based classification. It contains diverse fearure coding schemas for RNA, DNA and proteins, supports feature seletion, and integrates multiple classification methods.

2 Installation

2.1 Requirements

BioSeqClass employs some external programs to extract biological properties and use other R packages to build classification model:

1. BioSeqClass imported R packages are listed in table 1. These packages will be automatically installed when Biocalss is firstly loaded.

2. External programs are used to assist the performance of BioSeqClass (see table 2). Some programs are invoked via their web service, and some ones are needed to be installed at the local computer.

   Note: You do not need to install programs listed in table 2, unless you will use the related function in BioSeqClass.

2.2 Installation

The biocLite script is used to install PAnnBuilder from within R:

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("BioSeqClass")
```

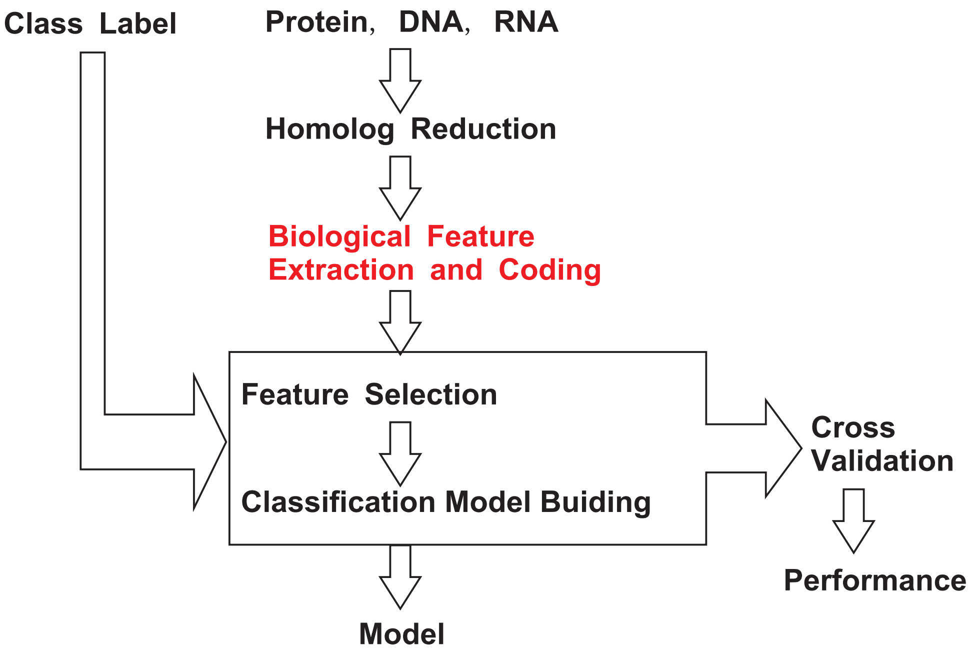

Figure 1: Workflow for Biological Sequence based Classification.

Users also can use the installation script ”BioSeqClass.R” to download and install BioSeqClass package.

```
> source("http://www.biosino.org/download/BioSeqClass/BioSeqClass.R")
> BioSeqClass()
```

Load package:

```
> library(BioSeqClass)
```

Note: Web Connection is needed to install BioSeqClass and its required packages. All the codes in this vignette were tested in R 2.8.0 and 2.9.0, thus the latest R version is recommended.

3 Function description

3.1 Homolog Reduction

Homologous sequences in training/testing data may lead to overestimation of prediction accuracy. Therefore, the first step for sequence based classification and prediction is homolog reduction based on sequence similarity. Taking computation complexity and similarity restriction into consideration, homolog reductions for full-length sequences and fragment sequences are different. We have designed different functions to deal with them, respectively (see table 3).

- `hr` - It employs `cdhitHR` and `aligndisHR` to filter homolog sequences by sequence similarity. `cdhitHR` is designed to filter full-length protein or gene sequences. `aligndisHR` is designed for aligned sequences with equal length.

- `cdhitHR` - It uses cd-hit program to do homolog reduction (”formatdb” and ”blastall” are required for running cd-hit program). CD-HIT is a program for clustering large protein database at high sequence identity threshold (Li and Godzik, 2006).

- `aligndisHR` - It uses the number of different residues to do homolog reduction (D et al., 2008). The algorithm proceeds in a stepwise manner by first eliminating sequences that were different from another in exactly 1 position. Elimination proceeds one peptide at a time; Re-evaluate after each peptide is removed. Once no further homologs of distance 1 remain, homologs of distance 2 are eliminated, and so forth until identity between all peptides are less than given cutoff.

3.2 Feature Extraction and Numerical Coding

3.2.1 Biological sequences

RNA, DNA and protein are three kinds of basic biological sequences. RNA and DNA are composed of bases, while proteins are composed of amino acids. Furthermore, amino acids have different physical-chemical properties, which were used to divide amino acids into different groups. These elements and groups are basic objects for feature extraction (see table 4).

3.2.2 Feature Coding

Feature coding means to extract features from sequences and convert them into numerical values. The frequently used features are the basic elements of sequence (bases, amino acids), physical-chemical properties, secondary structures, and so on. There are many methods to convert features to numerical values. The simplest is the composition of element. But more sophisticated conversions are preferrable for achieving better distinguishing power. Previous studies have shown that feature coding is the key point for the accuracy of classification and prediction. Here we have summarized various feature coding methods used in published papers and carried out them in BioSeqClass (see table 5). These functions will allow more diverse choices of coding strategies and accelerate the feature coding process. We also provide a function `featureEvaluate` to test the performance of models with different feature coding schemes and different classification algorithms.

3.3 Feature Selection

Features are important for the accuracy of prediction model. However, it does not mean that the more the better. Computation time is usually increased with the increase of number of features. Conflictive features would even reduce the accuracy. Therefore, suitable feature selection is needed for better prediction performance and less computation cost. We provided two functions for feature selection (see table 6).

3.4 Model Building and Performance Evaluation

Besides features, classification method is another factor that influences classification. Different cases may have different perference over classification methods. Multiple classification methods are integrated and available in BioSeqClass (see table 7). To evaluate and compare classification models, performance assessment is done for each model, including precision, sensitivity, specificity, accuracy, and matthews correlation coefficient.

Table 1: Imported R packages.

Existing R Package	**Functions used by BioSeqClass**
Biostrings	readFASTA, writeFASTA
e1071	svm
ipred	bagging
klaR	svmlight, NaiveBayes
randomForest	randomForest
class	knn
tree	tree
nnet	nnet
rpart	rpart
party	ctree
foreign	write.arff
Biobase	addVigs2WinMenu

4 Examples

To illustrate the use of BioSeqClass, lysine acetylation site prediction is taken as an example.

1. Suppose the original data are protein FASTA sequences and lysine acetylation sites. You can use `getTrain` to extract the flanking peptides of acetylation sites as positive dataset, and filter these peptides based on sequence identity. Lysine without acetylation annotation are regarded as negative dataset, and are filtered like the positive dataset. Considering the computational time, only 20 positive data are used as examples in the following codes.

```
> library(BioSeqClass)
> file = file.path(.path.package("BioSeqClass"), "example", "acetylation_K.fasta")
> posfile = file.path(.path.package("BioSeqClass"), "example",
+     "acetylation_K.site")
> posfile1 = tempfile()
> write.table(read.table(posfile, sep = "\t", header = F)[1:20,
+     ], posfile1, sep = "\t", quote = F, row.names = F, col.names = F)
> seqList = getTrain(file, posfile1, aa = "K", w = 7, identity = 0.4)
```

```
[1] "Positive Site: 18"
[1] "Positive Protein: 9"
[1] "Positive Site After Homolog Reduction: 16"
[1] "Positive Protein After Homolog Reduction: 9"
[1] "Negative Site: 215"
[1] "Negative Protein: 9"
```

Table 2: Invoked External Programs.

External Program	Description	Related BioSeqClass Function	Need Installed?	Ref
cd-hit	a program for clustering large protein database at high sequence identity threshold	`cdhitHR`	Yes	(Li and Godzik, 2006)
blastpgp	PSI-BLAST (Position-Specific Iterated BLAST) for capturing the conservation pattern	`featurePSSM`	Yes	(Altschul et al., 1997)
SVMlight	support vector machine	`classifyModelSVMLIGHT`	No	(T, 1999)
DSSP	a database of secondary structure assignments for protein entries in the Protein Data Bank (PDB)	`getDSSP`	No	(Kabsch and Sander, 1983)
Proteus2	predict secondary structure	`predictPROTEUS`	No	(S et al., 2006)
HMMER	predict domains with hmmpfam using models of Pfam database	`predictPFAM`	No	(Eddy, 1998)

Table 3: Summary Table for Homolog Reduction Functions.

Function	Description	Ref
`hr`	employ `cdhitHR` and `aligndisHR` to do homolog reduction	
`cdhitHR`	invoke cd-hit to cluster sequences	(Li and Godzik, 2006)
`aligndisHR`	calculated identity of aligned sequences	(D et al., 2008)

Table 4: Summary Table for Base and Amino Acid Groups.

Function	Description	Ref
`elements`	basic elements of biological sequence	
`aaClass`	amino acids groups depend on their physical-chemical properties: hydrophobicity, normalized Van der Waals volume, polarizability, polarity, and so on	(CS et al., 2006)

Table 5: Summary table for Feature Coding Functions.

Type	Function	Feature Coding Scheme	Ref
DNA, RNA, or protein	`featureBinary`	use 0-1 vector to code each element	(Jia et al., 2006; Chen et al., 2006)
	`featureCTD`	numeric vector for the composition, transition and distribution of properties	(Cai et al., 2003)
	`featureFragmentComposition`	numeric vector for the frequency of k-mer sequence fragment	(CS et al., 2004; RY et al., 2002)
	`featureGapPairComposition`	numeric vector for the frequency of g-spaced element pair	(CS et al., 2006)
	`featureCKSAAP`	integer vector for the number of k-spaced element pair (k cycled from 0 to g)	(Chen et al., 2008)
protein	`featureHydro`	hydrophobic effect	(A and R, 1988; MJ et al., 2000)
	`featureACH`	average cumulative hydrophobicity over a sliding window	(Zhang et al., 2008; Kurgan et al., 2007)
	`featureAAindex`	numeric vector measuring the physicochemical and biochemical properties based on AAindex database	(S et al., 1999; Cai and Lu, 2008)
	`featureACI`	numeric vector measuring the average cumulative properties in AAindex	
	`featureACF`	numeric vector measuring the Auto Correlation Function (ACF) of properties in AAindex	(Zhang et al., 1998; Liu and Chou, 1998)
	`featurePseudoAAComp`	numeric vector for the pseudo amino acid composition proposed by Chou,K.C.	(Chou, 2001)
	`featurePSSM`	numeric vector for the normalized position-specific score of PSSM generated by PSI-BLAST	(Zhang et al., 2005; Chung-Tsai Su and Ou, 2006; S and A, 2005; Zhang et al., 2008)
	`featureDOMAIN`	vector for the number of domain. Domains can be obtained by 'predictPFAM' function.	(Jia et al., 2007)
	`featureSSC`	coding for secondary structure of protein. Secondary structure can be got by 'predictPROTEUS' or 'getDSSP'.	(Zheng and Kurgan, 2008)
DNA or RNA	`featureBDNAVIDEO`	Conformational and physicochemical DNA features from B-DNA-VIDEO database	(v. Ponomarenko et al., 1999)
	`featureDIPRODB`	conformational and thermodynamic dinucleotide properties from DiProDB database (http://diprodb.fli-leibniz.de)	(Friedel et al., 2009)

Table 6: Summary Table for Feature Selection Functions.

Function	**Description**	**Ref**
`selectWeka`	feature selction by methods in WEKA	(Witten and Fran, 2005)
`selectFFS`	feature forword selction based on the performance of classification model	(Cai and Lu, 2008)
`classify`	build and test model with cross validation, also support feature selection by envoking WEKA	

Table 7: Summary Table for Classification Methods.

Function	**Description**	**Depended R package**
`classifyModelLIBSVM`	support vector machine by LIBSVM	*e1071*
`classifyModelSVMLIGHT`	support vector machine by SVM-light	*klaR*
`classifyModelNB`	naive bayes	*klaR*
`classifyModelRF`	random forest	*randomForest*
`classifyModelKNN`	k-nearest neighbor	*class*
`classifyModelTree`	tree model	*tree*
`classifyModelNNET`	neural net algorithm	*VR*
`classifyModelRPART`	recursive partitioning trees	*rpart*
`classifyModelCTREE`	conditional inference trees	*party*
`classifyModelCTREELIBSVM`	combine conditional inference trees and support vector machine	*party*, *e1071*
`classifyModelBAG`	bagging method	*ipred*

```
[1] "Negative Site After Homolog Reduction and Random Selection: 16"
[1] "Negative Protein After Homolog Reduction and Random Selection: 7"
```

2. If the original data are non-redundant positive/negative peptides. We directly read the data into R and assign class labels for them.

```
> tmpDir = file.path(.path.package("BioSeqClass"), "example")
> tmpFile1 = file.path(tmpDir, "acetylation_K.pos40.pep")
> tmpFile2 = file.path(tmpDir, "acetylation_K.neg40.pep")
> posSeq = as.matrix(read.csv(tmpFile1, header = F, sep = "\t",
+     row.names = 1))[, 1]
> negSeq = as.matrix(read.csv(tmpFile2, header = F, sep = "\t",
+     row.names = 1))[, 1]
> seq = c(posSeq, negSeq)
> classLable = c(rep("+1", length(posSeq)), rep("-1", length(negSeq)))
> length(seq)
```

```
[1] 1200
```

3. Once you have positive/negative datasets, you can code them to numeric vectors by functions listed in table 5. Function `featureBinary` and `featureGapPairComposition` are taken as examples of different coding methods, which use binary 0-1 coding and the composition of gapped amino acid pair, respectively. Other functions can be used in the same way.

```
> feature1 = featureBinary(seq, elements("aminoacid"))
> dim(feature1)
```

```
[1] 1200  300
```

```
> feature2 = featureGapPairComposition(seq, 0, elements("aminoacid"))
> dim(feature2)
```

```
[1] 1200  400
```

4. `classify` is used to build classification model under cross validation. It also supports feature selection by invoking WEKA. Models built with selected features usually can obtain higher accuracy. In the following codes, two models are built by `classify`. The 1st classification model 'LIBSVM_CV5' is built by support vector machine with linear kernel and get an accuracy of 0.56 under 5-fold cross validation. The 2nd classification model 'FS_LIBSVM_CV5' is also built by support vector machine with linear kernel, but a feature selection method called ”CfsSubsetEval” is used before building model. Thus the 2nd model using feature selection achieves an higher accuracy of 0.62 than the 1st model using all features.

```
> data = data.frame(feature1, classLable)
> LIBSVM_CV5 = classify(data, classifyMethod = "libsvm", cv = 5,
+     svm.kernel = "linear", svm.scale = F)
> LIBSVM_CV5[["totalPerformance"]]

         tp          tn          fp          fn         prc          sn
337.0000000 337.0000000 263.0000000 263.0000000   0.5599397   0.5616667
         sp         acc         mcc          pc
  0.5616667   0.5616667   0.1243355   0.3918323

> FS_LIBSVM_CV5 = classify(data, classifyMethod = "libsvm", cv = 5,
+     evaluator = "CfsSubsetEval", search = "BestFirst", svm.kernel = "linear",
+     svm.scale = F)
> FS_LIBSVM_CV5[["totalPerformance"]]

         tp          tn          fp          fn         prc          sn
258.0000000 488.0000000 112.0000000 342.0000000   0.6924855   0.4300000
         sp         acc         mcc          pc
  0.8133333   0.6216667   0.2621407   0.3651539

> colnames(data)[FS_LIBSVM_CV5$features[[1]]]

 [1] "BIN.1_A"  "BIN.2_R"  "BIN.3_K"  "BIN.4_K"  "BIN.5_K"  "BIN.5_N"
 [7] "BIN.6_N"  "BIN.6_G"  "BIN.6_T"  "BIN.6_P"  "BIN.7_P"  "BIN.9_W"
[13] "BIN.9_H"  "BIN.9_Y"  "BIN.9_L"  "BIN.10_Q" "BIN.11_W" "BIN.11_V"
[19] "BIN.12_K" "BIN.12_F" "BIN.13_K" "BIN.13_I" "BIN.13_M" "BIN.14_A"
[25] "BIN.15_K" "BIN.15_E" "BIN.15_A" "BIN.15_T"
```

5. Different feature coding methods usually might result in different prediction performance. featureEvaluate can be used to test multiple feature coding methods. Figure 2 shows the 3D plot of prediction accuracy varied with feature coding functions and parameters. It can be generated by employing featureEvaluate as follows (Note: It may be time consuming!):

```
> fileName = tempfile()
> testFeatureSet = featureEvaluate(seq, classLable, fileName, cv = 5,
+     ele.type = "aminoacid", featureMethod = c("Binary", "GapPairComposition"),
+     classifyMethod = "libsvm", group = c("aaH", "aaV", "aaZ",
+         "aaP", "aaF", "aaS", "aaE"), g = 0, hydro.methods = c("kpm",
+         "SARAH1"), hydro.indexs = c("hydroE", "hydroF", "hydroC"))
> summary = read.csv(fileName, sep = "\t", header = T)
> if (!require("scatterplot3d")) {
+     install.packages("scatterplot3d", repos = "http://stat.ethz.ch/CRAN",
+         quiet = T)
+ }
> require("scatterplot3d")
> tmp1 = summary[, "Feature_Function"]
> tmp2 = as.factor(sapply(as.vector(summary[, "Feature_Parameter"]),
+     function(x) {
+         unlist(strsplit(x, split = "; "))[1]
+     }))
> testResult = data.frame(as.integer(tmp2), as.integer(tmp1), summary[,
+     "acc"])
> s3d = scatterplot3d(testResult, color = c("red", "blue")[testResult[,
+     2]], pch = 19, xlab = "Parameter", ylab = "Feature Coding",
+     zlab = "Accuracy", lab = c(9, 3, 7), x.ticklabs = gsub("class: ",
+         "", sort(unique(tmp2))), type = "h", ylim = c(0, 3),
+     y.margin.add = 2.5, y.ticklabs = c("", gsub("feature", "",
+         sort(unique(tmp1))), ""))
```

6. Features from multiple functions can be combined and re-selected to increase the prediction accuracy. In the following code chunk, the first three feature sets from 'testFeatureSet' are combined together ('testFeatureSet' is generated in the aforementioned codes by `featureEvaluate`). Then feature selection functions (`classify` and `selectFFS`) can be employed to selecte features. (`classify` has been illustrated in the aforementioned code chunk. Thus `selectFFS` is used here to do feature forward selection to select a subset with maximum prediction accuracy (Note: It may be time consuming!). The process of feature selection and the increasing accuracy are shown in Figure 3.

```
> feature.index = 1:3
> tmp <- testFeatureSet[[1]]$data
> tmp1 <- testFeatureSet[[feature.index[1]]]$model
> colnames(tmp) <- paste(tmp1["Feature_Function"], tmp1["Feature_Parameter"],
+     colnames(tmp), sep = " ; ")
> data <- tmp[, -ncol(tmp)]
> for (i in 2:length(feature.index)) {
+     tmp <- testFeatureSet[[feature.index[i]]]$data
+     tmp1 <- testFeatureSet[[feature.index[i]]]$model
+     colnames(tmp) <- paste(tmp1["Feature_Function"], tmp1["Feature_Parameter"],
+         colnames(tmp), sep = " ; ")
+     data <- data.frame(data, tmp[, -ncol(tmp)])
+ }
> name <- colnames(data)
> data <- data.frame(data, tmp[, ncol(tmp)])
> combineFeatureResult = selectFFS(data, accCutoff = 0.005, classifyMethod = "knn"
+     cv = 5)
> tmp = sapply(combineFeatureResult, function(x) {
```

```
+     c(length(x$features), x$performance["acc"])
+ })
> plot(tmp[1, ], tmp[2, ], xlab = "Feature Number", ylab = "Accuracy",
+     , pch = 19)
> lines(tmp[1, ], tmp[2, ])
```

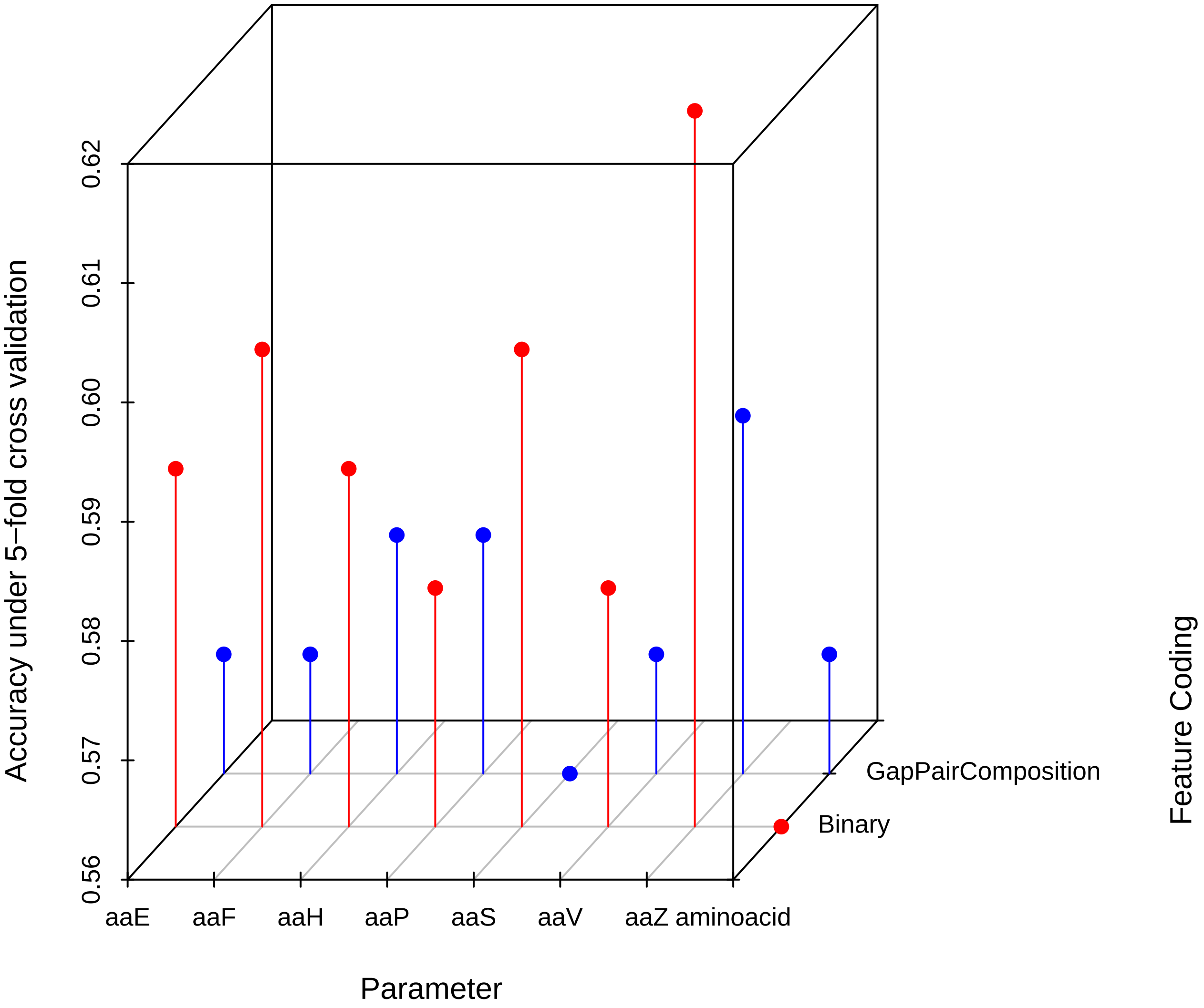

Figure 2: Result of `featureEvaluate`.

5 Session Information

This vignette was generated using the following package versions:

```
R version 2.13.0 (2011-04-13)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=C              LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
```

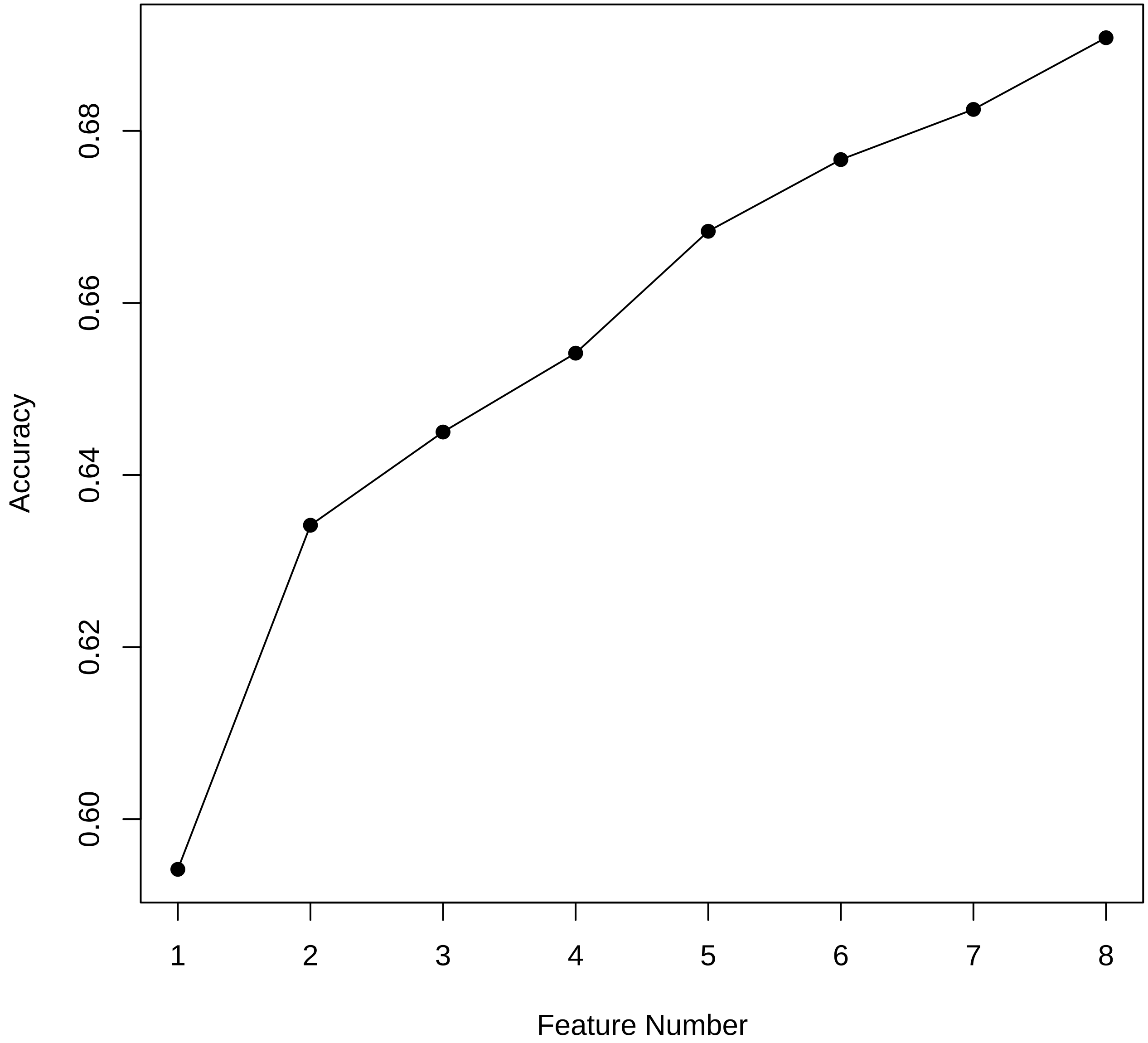

Figure 3: Result of `selectFFS`.

```
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats4    splines   stats     graphics  grDevices utils     datasets
[8] methods   base

other attached packages:
[1] BioSeqClass_1.10.0 modeltools_0.2-17  mvtnorm_0.9-96     nnet_7.3-1
[5] class_7.3-3        survival_2.36-5    mlbench_2.1-0      MASS_7.3-12
[9] rpart_3.1-49

loaded via a namespace (and not attached):
 [1] Biobase_2.12.0     Biostrings_2.20.0  IRanges_1.10.0     coin_1.0-18
 [5] colorspace_1.0-1   e1071_1.5-25       foreign_0.8-43     ipred_0.8-11
 [9] klaR_0.6-5         party_0.9-99991    randomForest_4.6-2 tools_2.13.0
[13] tree_1.0-28
```

References

Radzicka A and Wolfenden R. Comparing the polarities of the amino acids: Side-chain distribution coefficients between the vapor phase, cyclohexane, 1-octanol, and neutral aqueous solution. *Biochemistry*, 27:1664–1670, 1988.

S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Res*, 25:3389–3402, 1997.

C.Z. Cai, L.Y. Han, Z.L. Ji, X. Chen, and Y.Z. Chen. Svm-prot: web-based support vector machine software for functional classification of a protein from its primary sequence. *Nucleic Acids Research*, 31(13):3692ĺC3697, 2003.

Yu-Dong Cai and Lin Lu. Predicting n-terminal acetylation based on feature selection method. *Biochemical and Biophysical Research Communications*, 372(4):862–865, 2008.

Hu Chen, Yu Xue, Ni Huang, Xuebiao Yao, and Zhirong Sun. Memo: a web tool for prediction of protein methylation modifications. *Protein Sci*, 34:249–53, 2006.

Yong-Zi Chen, Yu-Rong Tang, Zhi-Ya Sheng, and Ziding Zhang. Prediction of mucin-type o-glycosylation sites in mammalian proteins using the composition of k-spaced amino acid pairs. *BMC Bioinformatics*, 9:101, 2008.

K. C Chou. Prediction of protein cellular attributes using pseudo-amino acid composition. *Proteins*, 43(3):246–255, 2001.

Chien-Yu Chen Chung-Tsai Su and Yu-Yen Ou. Protein disorder prediction by condensed pssm considering propensity for order or disorder. *Bioinformatics*, 7:319, 2006.

Yu CS, Lin CJ, and Hwang JK. Predicting subcellular localization of proteins for gram-negative bacteria by support vector machines based on n-peptide compositions. *Protein Sci*, 13(5):1402–6, 2004.

Yu CS, Chen YC, Lu CH, and Hwang JK. Prediction of protein subcellular localization. *Proteins*, 64(3):643–51, 2006.

Schwartz D, Chou MF, and Church GM. Predicting protein post-translational modifications using meta-analysis of proteome-scale data sets. *Mol Cell Proteomics*, 2008.

S. R. Eddy. Profile hidden markov models. *Bioinformatics*, 14:755–763, 1998.

Maik Friedel, Swetlana Nikolajewa, Jurgen Suhne, and Thomas Wilhelm. Diprodb: a database for dinucleotide properties. *Bioinformatics*, 37:37–40, 2009.

Peilin Jia, Tieliu Shi, Yudong Cai, and Yixue Li. Demonstration of two novel methods for predicting functional sirna efficiency. *BMC Bioinformatics*, 7:271, 2006.

Peilin Jia, Ziliang Qian, Zhen Bin Zeng, Yudong Cai, and Yixue Li. Prediction of subcellular protein localization based on functional domain composition. *Biochem Biophys Res Commun,*, 357(2):366–370, 2007.

W. Kabsch and C Sander. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22(12):2577–637, 1983.

Lukasz A. Kurgan, Wojciech Stach, and Jishou Ruan. Novel scales based on hydrophobicity indices for secondary protein structure. *J. Theor. Biol.*, 248:354ÍC366, 2007.

Weizhong Li and Adam Godzik. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 19(1):155–6, 2006.

Wei-min Liu and Kuo-Chen Chou. Prediction of protein structural classes by modified mahalanobis discriminant algorithm. *Journal of Protein Chemistry*, 17(3):209–217, 1998.

Korenberg MJ, David R, Hunter IW, and Solomon JE. Automatic classification of protein sequences into structure/function groups via parallel cascade identification: a feasibility study. *Ann Biomed*, 28(7):803–811, 2000.

Luo RY, Feng ZP, and Liu JK. Prediction of protein structural class by amino acid and polypeptide composition. *Eur J Biochem*, 269(17):4219–4225, 2002.

Ahmad S and Sarai A. Pssm-based prediction of dna binding sites in proteins. *BMC Bioinformatics*, 6:33, 2005.

Kawashima S, Ogata H, and Kanehisa M. Aaindex: amino acid index database. *Nucleic Acids Res*, 27:368–369, 1999.

Montgomerie S, Sundararaj S, Gallin WJ, and Wishart DS. Improving the accuracy of protein secondary structure prediction using structural alignment. *BMC Bioinformatics*, 14:301, 2006.

Joachims T. Making large-scale svm learning practical. *MIT Press*, pages 169–184, 1999.

Julia v. Ponomarenko, Mikhail P. Ponomarenko, Anatoly S. Frolvo, Denies G. Vorobyev, G.Christian Overton, and NIkolay A. Kolchanov. Conformational and physicochemical dna features specific for transcription factor binding sites. *Bioinformatics*, 15:654–668, 1999.

Ian H. Witten and Eibe Fran. Data mining: Practical machine learning tools and techniques. *Morgan Kaufmann, San Francisco*, 2005.

C. Zhang, Z. Lin, Z. Zhang, and M. Yan. Prediction of the helix/strand content of globular proteins based on their primary sequences. *Protein Eng*, 11(11):971–979, 1998.

Qidong Zhang, Sukjoon Yoon, and William J. Welsh. Improved method for predicting beta-turn using support vector machine. *Bioinformatics*, 21(10):2370–2374, 2005.

Tuo Zhang, Ke Chen Hua Zhang, Shiyi Shen, Jishou Ruan, and Lukasz Kurgan. Accurate sequence-based prediction of catalytic residues. *Bioinformatics*, 24(20):329ĺC2338, 2008.

Ce Zheng and Lukasz Kurgan. Prediction of beta-turns at over 80predicted secondary structures and multiple alignments. *BMC Bioinformatics*, 9:430, 2008.