G2PML: tutorial on basic usage of the tool

Juan A. Botía

25/04/2019

Introduction

G2PML is set of datasets and software to study monogenic diseases. We study the diseases through the genes already discovered to be associated to the disease in the Mendelian way (i.e. one mutation, one case). These genes form what is known as a panel of genes.

Our approach to the disease is by describing genes in terms of (a) genetic constraint related properties, (b) genomic related properties and (c) transcriptomics.

With G2PML and your panel P={g1,g2,...,gn} of choice, where gi is the i-th gene, you can

- Perform an analysis on which features are relevant to distinguish those genes from the rest
- Create prediction models to predict new genes from the set of P genes and annotate and score the predictions.
- Enlarge your DDBB of properties to incorporate new ones to enrich your analysis

In our studies, we have been using the PanelApp database of gene panels, from Genomics England, accesible at http://panelapp.genomicsengland.co.uk. In the package we incorporta functions to navigate the panels and access the genes. For example, in

```
library(G2PML)
panels = getPanelsFromPanelApp()
names(panels)
```

```
##  [1] "Name"               "Number_of_Regions"  "CurrentCreated"
##  [4] "Status"             "Number_of_STRs"     "Relevant_disorders"
##  [7] "CurrentVersion"     "DiseaseSubGroup"    "DiseaseGroup"
## [10] "Panel_Id"           "Number_of_Genes"    "PanelTypes"
```

```
dim(panels)
```

```
## [1] 173  12
```

```
panels$Name[1:10]
```

```
##  [1] "Adult solid tumours for rare disease"
##  [2] "Amelogenesis imperfecta"
##  [3] "Amyotrophic lateral sclerosis/motor neuron disease"
##  [4] "Anophthalmia or microphthalmia"
##  [5] "Arrhythmogenic cardiomyopathy"
##  [6] "Arthrogryposis"
##  [7] "Atypical haemolytic uraemic syndrome"
##  [8] "Auditory Neuropathy Spectrum Disorder"
##  [9] "Autosomal recessive congenital ichthyosis"
## [10] "Beckwith-Wiedemann syndrome (BWS) and other congenital overgrowth disorders"
```

We can access 173 gene panels and their genes. For example, we can get the genes for monogenic forms of PD as follows

```
genes = getGenesFromPanelApp (disorder="Neurology and neurodevelopmental disorders",
                      panel="Parkinson Disease and Complex Parkinsonism",
                      color="green")
genes
```

```
##  [1] "PRKN"     "ATP13A2"  "ATP1A3"   "C19orf12" "CSF1R"    "DCTN1"
##  [7] "DNAJC6"   "FBXO7"    "FTL"      "GBA"      "GCH1"     "GRN"
## [13] "LRRK2"    "LYST"     "MAPT"     "OPA3"     "PANK2"    "PARK7"
## [19] "PINK1"    "PLA2G6"   "PRKRA"    "RAB39B"   "SLC30A10" "SLC39A14"
## [25] "SLC6A3"   "SNCA"     "SPG11"    "SPR"      "SYNJ1"    "TH"
## [31] "TUBB4A"   "VPS13A"   "VPS35"    "WDR45"
```

We'll take those genes as an example on how to use GP2ML to study the phenotype.

Relevant features on your gene panel

At this point, we have ready our gene list to study. And we want to know what makes it different this list of genes from the rest of genes in the genome (note we only consider for now protein coding genes as many of the features we use to desdribe them just only make sense for that gene biotype).

Note that we have many different properties to describe our genes. To quickly get a hint on how these genes are described, we can do

```
genedata = G2PML::fromGenes2MLData(genes=genes,which.controls="allgenome")
```

```
## ALLGENOME mode: We have to remove 34 disease genes from the all genome controls
## We have to remove 52 genes RTF2 MTREX TUT7 RAB5IF PPP1R2C  because we don't know them
## Removing columns for attributes DPI and DSI and ESTcount and constitutiveexons
```

```
dim(genedata)
```

```
## [1] 17636   161
```

By simply doing that, we have created an annotation dataset for our genes (and also for the rest of genes in the genome), i.e. 17635 genes including the PD ones, with 161 features of interest. A quick look at the features we use ...

```
str(genedata[,1:30])
```

```
## 'data.frame':    17636 obs. of  30 variables:
##  $ gene                                : chr  "PRKN" "ATP13A2" "ATP1A3" "C19orf12" ...
##  $ GeneLength                          : num  1380351 25970 30915 16903 60081 ...
##  $ TranscriptCount                     : num  11 15 13 9 9 ...
##  $ CountsOverlap                       : int  3 1 0 0 2 2 3 0 0 1 ...
##  $ NumJunctions                        : int  65 145 120 13 65 287 76 51 3 66 ...
##  $ IntronicLength                      : int  1375858 20744 25870 11424 54953 20527 160766 15642 703
##  $ GCcontent                           : num  40.4 57.2 56.1 50.4 48.1 ...
##  $ String                              : num  1.61 5 3 0 8 ...
##  $ LoFTool                             : num  0.46028 0.0289 0.00296 0.041 0.15 ...
##  $ EvoTol                              : num  50.31 1.16 15.59 11.61 19.18 ...
##  $ RVIS                                : num  1.035 0.975 0.558 0.986 0.942 ...
##  $ pAD                                 : num  0.436 0.26 1 0.995 0.787 ...
##  $ pAR                                 : num  0.564 0.74 0 0.005 0.213 ...
##  $ gnomadpLI                           : num  0.356 0.125 1 0.466 0.356 ...
##  $ gnomadpRec                          : num  0.6376 1 0.0333 0.6525 0.6376 ...
##  $ gnomadpNull                         : num  3.13e-01 4.48e-02 5.15e-05 2.01e-01 3.13e-01 ...
```

```
##  $ gnomadpMiss                          : num  1.067 1.48 6.589 0.531 1.595 ...
##  $ gnomadOELoF                          : num  0.6256 0.4392 0.0414 0.4902 0.6256 ...
##  $ gnomadOEMiss                         : num  0.908 0.918 0.292 1.009 0.922 ...
##  $ RankedMMSpecificRankMMAdiposeSub     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ RankedMMSpecificRankMMAdiposeVisceral: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ RankedMMSpecificRankMMAdrenalGland   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ RankedMMSpecificRankMMArteryAorta    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ RankedMMSpecificRankMMArteryCoronary : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ RankedMMSpecificRankMMArteryTibial   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ RankedMMSpecificRankMMAmygdala       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ RankedMMSpecificRankMMAntCingCortex  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ RankedMMSpecificRankMMCaudate        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ RankedMMSpecificRankMMCerebHemisphere: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ RankedMMSpecificRankMMCerebellum     : int  0 0 0 0 0 0 0 0 0 0 ...
```

And we see we have the gene name, and many other features of interest that we describe elsewhere. But basically, between `GeneLength` and `GCcontent` we have genomic properties, `String` gives an indication on how active (or sticky) is the gene product (i.e. the protein). Then we have from `LoFTool` to `gnomadOEMiss` attributes that try to describe how resistant or fragile the gene is to mutation. The rest of properties are related with the expression and co-expression in GTEx 47 tissues.

Finally, we have

```
table(genedata$condition)
```

```
##
##    Disease Nondisease
##         34      17602
```

that helps us distinguish between our genes and the rest.

Now that we have this, we can perform an analysis on what are the most relevant features to distinguish, from a machine learning perspective, between your genes and the rest. For that we can perform a Caret based feature selection analysis. But we have to take into account that highly inbalance in the dataset, i.e. in the case of PD genes, we have 517 non panel genes for each of the genes in the panel. Also note that we expect that some of those genes not in the panel should also be there. And that is precissely what we aim for: to discover such genes.

We can perform a feature selection analysis simply doing this (it will take a while):

```
fspd = featureSelection(genes=genes,k=10,repeats=40,controls="allgenome",trnProp=0.9)
```

By using Caret and recursive feature subset eliminination strategy http://topepo.github.io/caret/recursive-feature-elimination.html averaged many times on perfectly balanced subproblems (i.e. 50% PD genes, 50% rest of protein coding genes) we will get a result set. It is important to describe the algorithm so we can appropriately understand results

- Step 1. Repeat the following steps for `repeats` times
  - Let `ctrl` be the control object resulting from a call to `caret::rfeControl` with random forest algorithms for feature selection, evaluating with cross-validation and using `k` folds
  - Create a training dataset `trn` using as positive examples `trnProp` proportion of the panel genes, and as negative examples, exactly the same number of genes randomly samples from the control genes
  - Call `caret::rfe` with the training data and `ctrl` and obtain the optimum variables for that experiment
  - Obtain the importance and sign of each variable as its coefficient in the regression model
- Step 2. Return a list of the result objects from each step in the loop

Let us suppose we have it saved and we load it now

```
fspd = readRDS("~/Dropbox/KCL/workspace/G2PML/inst/g2pml/Parkinson_Disease_and_Complex_Parkinsonism.fs.r
```

```
## Warning: replacing previous import 'ggplot2::empty' by 'plyr::empty' when
## loading 'caret'
```

```
metadata = G2PML::getVarsMetaDataFromFS(fspd,r=0.4)
```

```
## Working now with Unknown
```

```
eff = order(as.numeric(metadata$meaneffects[,"meaneffect"]))
metadata$meaneffects[,"meaneffect"][eff]
```

```
##           ExprSpecificFCortex                     gnomadpLI
##          "-0.300328945335776"          "-0.298463167912373"
##      ExprSpecificAdrenalGland                        String
##          "-0.258243695475316"          "-0.185526933091989"
##   ExprSpecificSubstantianigra    ExprSpecificMuscleSkeletal
##          "-0.178792394506871"          "-0.177154291999409"
##    AdjSpecificAdjColonSigmoid                      LoFTool
##          "-0.164954367083234"          "-0.148451430862964"
##      ExprSpecificHypothalamus                  GeneLength
##          "-0.141897873033249"          "-0.137344584422352"
##             ExprSpecificLiver                 gnomadOELoF
##          "-0.122230605877312"         "-0.0267538849693904"
##                 NumJunctions        ExprSpecificWholeBlood
##         "-0.0120114332543787"          "0.0110227654979671"
##                  gnomadpMiss                        RVIS
##           "0.054961938240615"          "0.0552493714660945"
##            ExprSpecificCortex                 gnomadOEMiss
##          "0.0697060202594493"          "0.0712383162283031"
##  ExprSpecificCellsLymphocytes                   GCcontent
##          "0.0714680772394238"           "0.126443651737662"
##                       EvoTol   ExprSpecificCerebHemisphere
##           "0.170115227227838"           "0.189256662730315"
##                    gnomadpRec                  gnomadpNull
##           "0.199017866281845"           "0.202321949288084"
##                IntronicLength              TranscriptCount
##           "0.209814755444196"           "0.318129228300929"
## ExprSpecificCellsFirbroblasts            ExprSpecificTestis
##           "0.441980705334164"           "0.615581923246568"
```

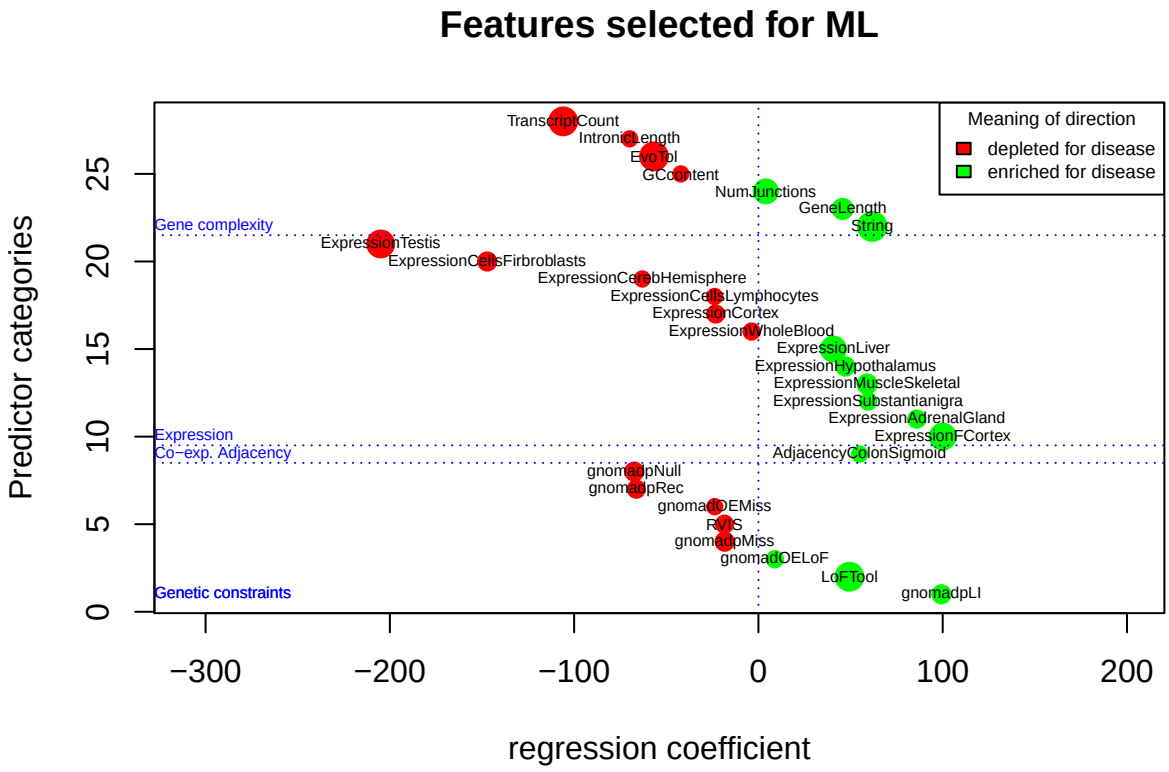
And from left to right we see the attributes that genes in the panel are enriched for higher values (left) and attributes (right) for which genes in the panel show lower varlues. Only those that show some statistical relevance are shown. We see that in terms of tissue and expression, Frontal cortex, adrenal gland, substantia nigra and scheletal muscle are relevant tissues. We also see that these genes lead to proteins with high interaction coefficients as found in String database, etc.

The r parameter is very important. It is the minimum proportion of times that a feature appeared selected as most relevant by Caret, out of the total number of iterations in the `repeat` variable. For example, in this case we repeated the same experiment 40 times, and we are asking to select all variables which appear more than 40% of those 40 times.

From that very same data, we can get a descriptive plot as follows

```
G2PML::featureSelectionPlot(fspd,r=0.4)
```

```
## Working now with FALSE
```



that shows how the gene panel is enriched for green feature values (higher values of x axes indicate more importance). For example, these genes are particularly expressed in Frontal Cortex and the Substantia Nigra. But they are also particularly abundant for PPI entries in the String database. It is the other way around for the red features. Note that the plot somewaht reflects the number of times the variable was selected by Caret as relevant out of the number of repeats through the blob's radius. Expression in Frontal Cortex is not only on of the features with strongest effect but also one of the most frequent within iterations. The same goes, on the opposite sense, for expression in testis and the number of transcripts at the genes.

Learning a model

The basic generation of an hypothesis from data in our system is based on Caret. However, we use Caret as a means to wrap any potentially useful learning algorithm in a common way. The basic hypothesis we create is from a perfectly balanced dataset, i.e. 50% genes from our panel, 50% genes from the rest of the genome. In consequence, to face a highly inbalanced problem, we have to convert that problem into many smaller and balanced problems, generate hypotheses for them and integrate all into an ensemble. Now, given one of those smaller learning problems, in order to get the best hypothesis, we employ a **tournament** based approach. This means that we use a pool of classification algorithms available in Caret (e.g. CART, C5.0, SVMs, random forest, LDA, knn) with the dataset, and keep the algorithm with the best test kappa. In order to compound an ensemble, we do this for each small learning problem. The hypotheses constructed in this way are integrated into the ensemble. Moreover, given that the genes in our panel are usually small, we have to repeat this process in different training/test folds to fight overfitting. In essence, the process would be the following, as coded in the `G2PML::ensembleLearnKFold()` function.

- Step 0. Let `nboot` be the max number of hypothesis that will form the final ensemble. Let `maxTrials` be the number of unsuccessful attempts of improving the overal Kappa statistic on the current ensemble. Let
- Step 1. For fold in 1..k do

  - Let $E \leftarrow \emptyset$ be the empty ensemble. Let $D = \{(x_i, y_i)\}$ be the learning data using as positive examples our panel genes and as negatives the rest of the genome
  - While $|E| < nboot$ and $i < maxTrials$
    * create d training data with 50% positive and 50% negative examples from D
    * For each algorithm in our pool of algorithms get the best hypothesis h^* amongst all. Add h^* to E if it improves evaluation error. If not, increment $maxTrials$
  - End while
- End For

Let us suppose we want to develop a model for the PD genes. We incorporate the results of the feature selection through `fspd`, we will use the default algorithms in the tournament (CART, C5.0, SVMs, random forest, LDA, knn), the number of basic models in the ensemble will be as much as 20, the number of folds in the cross-validation of the ensemble, $k = 5$ and the number of max trials without success before we stop growing the ensemble is also 5. As in the feature selection process, we use as controls the rest of the genome.

```
mypanel="Parkinson_Disease_and_Complex_Parkinsonism"
mymodel = ensembleLearnKFold(panel=mypanel,genes=genes,fs=fspd,nboot=20,
                             auto=T,k=5,maxTrials=5,controls="allgenome")
```

And we get

```
pdmodel = readRDS("~/Dropbox/KCL/workspace/G2PML/inst/g2pml/Parkinson_Disease_and_Complex_Parkinsonism.
names(pdmodel$eval)
```

```
## [1] "hits"           "hitsperfold"    "allpredictions" "bestq"
## [5] "qfold"          "finalpreds"     "kappa"
```

And now we study the most relevant fields. If we want to get the most reliable gene predictions, we do

```
print(pdmodel$eval$finalpreds)
```

```
##   [1] "ABCC8"    "ACACB"    "ACAN"     "ACLY"     "ACTB"     "ACTG1"
##   [7] "ACTN4"    "ACVR1"    "ADCY4"    "ADCY7"    "AGT"      "AKAP13"
##  [13] "AKT2"     "AMPD3"    "ANAPC5"   "ANK3"     "APC"      "APLP2"
##  [19] "APP"      "ARNT"     "ATF2"     "ATM"      "ATP2A2"   "ATP2A3"
##  [25] "ATP2B1"   "ATP2B2"   "ATXN3"    "BRCA1"    "CACNA1D"  "CACNA1H"
##  [31] "CACNB2"   "CACNB3"   "CALB1"    "CALD1"    "CALM1"    "CAMK2B"
##  [37] "CAPN1"    "CD36"     "CD44"     "CFTR"     "CHD3"     "CHD4"
##  [43] "CHD7"     "CHD9"     "CHEK2"    "CLASP2"   "CNTNAP2"  "COL11A1"
##  [49] "COL12A1"  "COL13A1"  "COL14A1"  "COL16A1"  "COL1A1"   "COL1A2"
##  [55] "COL21A1"  "COL27A1"  "COL4A2"   "COL5A1"   "COL6A3"   "COL7A1"
##  [61] "COL9A1"   "CPS1"     "CPSF7"    "CSK"      "CSNK1A1"  "CSNK1D"
##  [67] "CTBP2"    "CTNNA1"   "CTNND2"   "DAG1"     "DCTN2"    "DDX5"
##  [73] "DHCR7"    "DLG1"     "DLG2"     "DNM1"     "DNM1L"    "DNM2"
##  [79] "DNMT1"    "DPYSL2"   "DTNB"     "DYNC1I1"  "ECE1"     "EGFR"
##  [85] "EIF4G1"   "EP300"    "EPS15"    "ERBB3"    "ERC1"     "ERCC2"
##  [91] "ESR1"     "EWSR1"    "EYA1"     "EYA4"     "F5"       "FANCA"
##  [97] "FBLN1"    "FGFR2"    "FGFR3"    "FHL2"     "FLNB"     "FLT4"
## [103] "FN1"      "FOXM1"    "FOXP1"    "FOXP2"    "FUS"      "FYN"
## [109] "GCK"      "GMPR2"    "GNAI2"    "GNAO1"    "GNB2"     "GRB10"
## [115] "GRIN1"    "GRM7"     "GSN"      "GUK1"     "HDAC5"    "HDAC7"
## [121] "HDAC9"    "HGF"      "HMGA2"    "HNF4A"    "HNRNPC"   "HNRNPD"
## [127] "HNRNPK"   "HNRNPL"   "HRAS"     "HSD17B4"  "HSF4"     "HSP90B1"
## [133] "IKBKB"    "ITGA6"    "ITGA7"    "ITGB1"    "ITGB4"    "ITGB5"
## [139] "JAG1"     "KCNC2"    "KCNH1"    "KCNQ5"    "KIF1A"    "KIFC3"
```

```
## [145] "KLK3"     "LHX6"     "LRP1"     "LRP5"     "LRRK1"    "MAP2K2"
## [151] "MAP2K4"   "MAPK10"   "MDM2"     "MECP2"    "MED15"    "MED24"
## [157] "MEF2C"    "MEIS2"    "MEN1"     "MFGE8"    "MME"      "MPP4"
## [163] "MST1R"    "MYB"      "MYD88"    "MYH10"    "MYH11"    "MYH14"
## [169] "MYH9"     "MYO18A"   "MYO5A"    "MYO5C"    "MYO6"     "NAT9"
## [175] "NCAM1"    "NEDD4"    "NF1"      "NF2"      "NFATC4"   "NOP2"
## [181] "NOTCH2"   "NR4A1"    "NR4A2"    "NR5A2"    "NRP2"     "NTRK1"
## [187] "NTRK2"    "NTRK3"    "NUMA1"    "NUMB"     "NUP85"    "OTUB1"
## [193] "PABPC1"   "PAFAH1B1" "PAX5"     "PAX8"     "PCBP2"    "PDE2A"
## [199] "PDE4D"    "PDE4DIP"  "PDGFRB"   "PFKM"     "PGF"      "PITX2"
## [205] "PKD1"     "PLA2G4C"  "PLCB1"    "PLCB2"    "PLCB4"    "PLEC"
## [211] "PML"      "PORCN"    "PPARA"    "PPM1B"    "PPP2R5C"  "PRKAG2"
## [217] "PRKCE"    "PRKCZ"    "PRKDC"    "PRMT5"    "PSEN1"    "PSEN2"
## [223] "PTCH1"    "PTK2B"    "PTPN6"    "PXN"      "RAP1B"    "RAPGEF4"
## [229] "RARA"     "RBM5"     "RBPJ"     "RGS3"     "RGS6"     "RNF41"
## [235] "RORA"     "RORC"     "RPL13A"   "RPL17"    "RPS20"    "RPS6KA2"
## [241] "RRAS2"    "RUNX1T1"  "RUNX2"    "RXFP1"    "RYR3"     "SAP30BP"
## [247] "SCN1A"    "SCN2A"    "SCN5A"    "SCN8A"    "SERPINE2" "SERPINF2"
## [253] "SERPING1" "SERPINH1" "SF3B1"    "SGK1"     "SKIL"     "SMAD1"
## [259] "SMAD2"    "SMAD3"    "SMARCD1"  "SNRPA1"   "SOS1"     "SPTAN1"
## [265] "SQSTM1"   "SRC"      "SREBF1"   "SRRM1"    "STAT1"    "STAT3"
## [271] "STAT6"    "STRN4"    "STX1A"    "STXBP1"   "SUMO1"    "SUN1"
## [277] "SUPT5H"   "TACC2"    "TARDBP"   "TBL1XR1"  "TCF12"    "TCF7"
## [283] "TCF7L2"   "TGFBR1"   "THRA"     "THRB"     "TLE3"     "TNFRSF1A"
## [289] "TNNT1"    "TNNT2"    "TNNT3"    "TNS1"     "TP53"     "TP63"
## [295] "TP73"     "TPM1"     "TSC1"     "TSC2"     "U2AF1"    "UBTF"
## [301] "UCHL5"    "UPF1"     "VAV1"     "VDR"      "VIM"      "VWF"
## [307] "XPO1"     "XRN1"     "YAP1"     "ZEB1"     "ZEB2"
```

and we get 311 genes that, given the omics features of our gene panel, should be associated to monogenic forms of PD when data arrives for that. The Kappa statistic for the ensemble on the test data are

```
pdmodel$eval$kappa
```

```
##     folds meankappa  maxkappa  minkappa
## 5.0000000 0.5268570 0.6581828 0.3847122
```

Which are pretty decent, given that Kappa values are in $[-1, 1]$ and values over 0.5 are reasonably good.