Readme for R scripts in '7Q10 records and basin characteristics for 224 basins in South Carolina, Georgia, and Alabama (2015)' data release

https://doi.org/10.5066/F7CR5S4T

scworland@usgs.gov

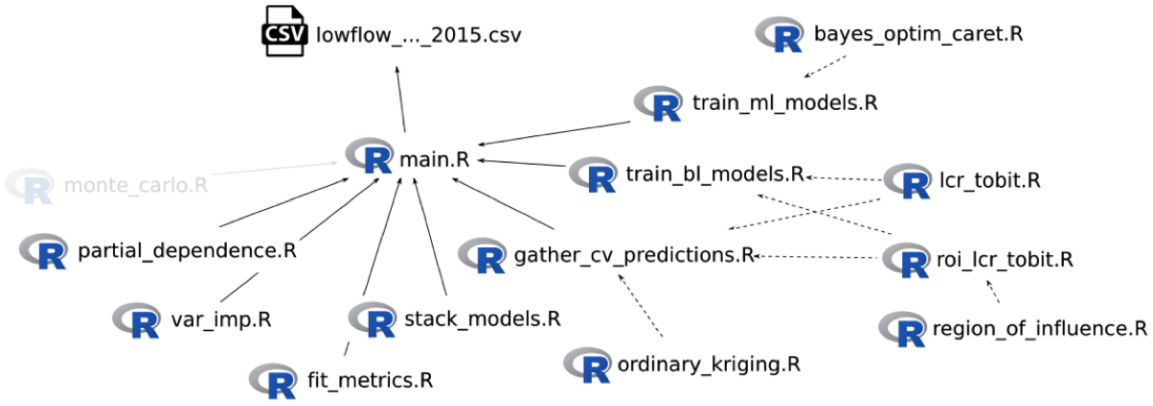
7/5/2017

Contents

Introduction 2
Descriptions of R scripts 3
Quick start 4
Extended analysis 7
    Description of train_ml_models() 7
    Description of train_bl_models() 8
R package citations 10

Introduction

This readme file is intended to serve as a navigation guide for the data release associated with the paper (citation information). The basic file contents and dependencies look like:



The arrows should be interpreted as "output dependency flow". The steps should be run in roughly clockwise order. Each primary step can be run directly with "main.R" file via custom R functions that have the same name as the othe R files in the figure above. Solid lines indicate primary functions that are sourced directly within the "main.R" file, and the dashed lines indicate other secondary functions that are sourced by the primary functions. For example, the "train_ml_models.R" file is sourced and run directly by "main.R", but "train_ml_models.R" is depedent on "bayes_optim_caret.R". The idea behind structuring the model archive in this way is that output dependencies (i.e., outputs of the functions) can be traced directly back to the input data csv file titled "lowflow_sc_ga_al_gagesII_2015.csv". The steps from the data file to the output should be completely reproducible[1] if functions are run in sequential order. The output of "stack_models.R" is a dataframe. This dataframe is included in the data release as "all_preds.csv". The monte_carlo.R script is greyed out because it was dropped from the analysis. Why it is included in the diagram is described in the table below.

Usage disclaimer: Although this software has been used by the USGS, no warranty, expressed or implied, is made by the USGS or the U.S. Government as to the accuracy and functioning of the program and related program material nor shall the fact of distribution constitute any such warranty, and no responsibility is assumed by the USGS in connection therewith.

Contact: Scott Worland, U.S. Geological Survey, Lower Mississippi-Gulf Water Science Center, 615-289-8319

Programming language citation: R Development Core Team (2017), R: A language and environment for statistical computing, R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0, http://www.R-project.org

R version: R version 3.4.0 (2017-04-21)

[1] With the exception of possible stochastic differences in model training. This is discussed in more detail in lower sections

Descriptions of R scripts

File name	File description	Dependencies
lowflow_sc_ga_al _gagesII_2015.csv	Main input datafile. Contains station ID, 7Q10 values, and basin characteristics.	no dependencies
train_ml_models.R	R script that sources primary and secondary function for analysis. This is the only R script that must be opened to run the analysis.	bayes_optim_caret.R
train_ml_models.R	Contains R function of same name that uses bayesian optimization to locate optimal hyperparameters for the machine learning models.	bayes_optim_caret.R
train_bl_models.R	Contains R function of same name that uses grid search to locate optimal hyperparameters for baseline regression models.	lcr_tobit.R, roi_lcr_tobit.R, region_of_influence.R
lcr_tobit.R	Contains R function of same name that calculates cross validated predictions from left censored Type I tobit regression model.	none
roi_lcr_tobit.R	Contains R function of same name that calculates cross validated predictions from region of influence left censored Type I tobit regression model.	region_of_influence.R
region_of_influence.R	Contains R function of same name that calculated euclidean distances between observations in predictor space.	none
gather_cv_predictions.R	Contains R function of same name that calculates the cross validated predictions for all models using the optimal hyperparameters.	ordinary_kriging.R, lcr_tobit.R, roi_lcr_tobit.R, region_of_influence.R
ordinary_kriging.R	Contains R function of same name that calculates predicted values using ordinary kriging.	none

stack_models.R	Contains R function of same name that ensembles the cross validated predictions from all other models.	none
fit_metrics.R	Contains R function of same name that calculates performance metrics using cross validated predictions from all models.	none
var_imp.R	Contains R function of same name that calculates the variable importance of machine learning models.	none
partial_dependence.R	Contains R function of same name that calculates partial dependence of most important predictor variables for several machine learning models.	none
monte_carlo.R	This was dropped from the analysis but described here only because earlier versions of the paper and data release had this component and we wanted to maintain a record of that.	none

Quick start

Although all the scripts are sourced by the "main.R"" script, most of them need not be run to recreate a bulk of the important results. The quickest way to get started is by opening the "main.R" script and setting up the work environment. The code below can be copy and pasted or run directly from the "main.R"" file. This includes installing pacakges and setting the working directory (note that the working directory is relative to where the files are stored, so a user needs to set their working directory after the files are downloaded).

```
# Install all packages required for entire analysis. The libraries needed for
# each function are loaded in the function script.

# add the USGS R repository (https://owi.usgs.gov/R/gran.html) to R profile so
# packages will download
rprofile_path = file.path(Sys.getenv("HOME"), ".Rprofile")
write('\noptions(repos=c(getOption(\'repos\'),
  CRAN=\'https://cloud.r-project.org\',
  USGS=\'https://owi.usgs.gov/R\'))\n',
    rprofile_path,
    append =  TRUE)

pkg_list <- c("dplyr","reshape2","randomForest","kknn","caret",
              "gbm","kernlab","Cubist","glmnet","devtools","AER",
```

```
              "leaps","doMC","ICEbox","geoR","smwrQW","smwrStats")

install.packages(pkg_list)

# install PUBAD package off of github
devtools::install_github("wfarmer-usgs/PUBAD")

# load libraries needed for main.R script
library(dplyr); library(PUBAD)

# set working directory
setwd("~/set/working/directoy")
```

The next step is to load the csv datafile and prepare the data as input for the functions. The only packages that are required for this step are `dplyr` and `PUBAD`.

```
# Load csv file: note, data should be in a "data" folder in working directory
data_full <- read.csv("data/lowflow_sc_ga_al_gagesII_2015.csv",header=T,na.strings = "-999") %>%
  setNames(tolower(names(.))) %>% # make column names lower case
  mutate(staid = paste0("0",staid)) # add leading zero

# Transform 7Q10 response variable
area <- data_full$drain_sqkm
ln7q10_da <- log((data_full$y7q10+0.001)/area)

# use functions from PUBAD to cull covariates
expVars <- data.frame(gages = data_full$staid) %>%
  getBasinChar()

# create model data using clean basin chars
model_data <- expVars$cleanBCs %>% # start with basin chars
  mutate(CLASS = as.integer(as.factor(CLASS))) %>% # change class to integer
  mutate_all(funs(as.numeric)) %>% # make everything numeric
  scale() %>% # convert to z-score: (x-mu)/sigma
  as.data.frame() %>% # make data frame
  setNames(tolower(names(.))) %>% # make sure colname are lower case
  mutate(y=ln7q10_da) %>% # add the transformed response variable from above
  select(y, class,lat_gage:aspect_eastness) # reorder column positions
```

The raw 7Q10s are divided by the drainage area of the basin and are log transformed prior to modelling. Because some of the 7Q10's are zero, a small, arbitrary constant of 0.001 was added to each 7Q10 value prior to taking the log transform. The data transformation for the baseline models is slightly different, and is explained in the "Description of train_bl_models" section below. After the data are loaded, the first step in the "main.R" is to find the optimal hyperparameters of for all of the models using random search, grid search, and bayesian optimization. This process will take over 12 hours for all of the models, so we provide the leave-one-out predicted values (output from "stack_models()" function) in the "all_preds.csv" file. You can go ahead and jump straight to the goodness of fit section in the "main.R" file starting at line 121.

```
# source fit_metrics function
source("scripts/fit_metrics.R")

# load cross validated predictions
all_preds <- read.csv("data/all_preds.csv")
```

```
# calculate model error
model_error <- fit_metrics(all_preds,data_full)
```

The final steps for this abbreviated version of the analysis is to calculate variable importance, partial dependence, and make some plots.

```
# calculate variable importance
source("scripts/var_imp.R")
var_imp_overall <- var_imp(model_data)

# calculate partial dependence
source("scripts/partial_dependence.R")
pdp_data <- partial_dependence(model_data,var_imp_overall)
```

Note that the variable importance and partial dependence don't actually require direct inputs from the other functions above. However, the model selections for the variable importance are based off pieces of the earlier analysis. The very last step is to create plots from the paper:

```
# generate list of  plots
source("scripts/make_plots.R")
plots <- make_plots(all_preds,model_error,var_imp_overall,pdp_data)

# display the plots
plots$rmse_vs_unitrmse
plots$pred_vs_obs
plots$error_decomp
plots$var_imp
plots$partial_dep
```

Extended analysis

This section includes more information about determining the hyperparameters of the models and the monte carlo analysis.

Description of train_ml_models()

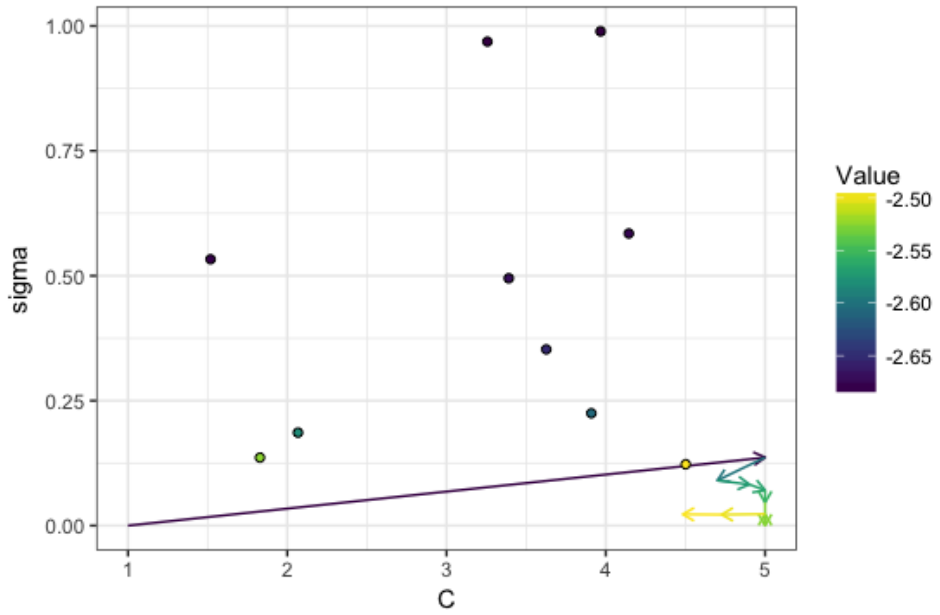
The process for the machine learning models is conceptually similar to the baseline models, but is much more computationally expensive. This basic pipeline involves (1) generating a bunch of points in hyperparameters and RMSE space to be used as substrate for a gaussian process model, and (2) use bayesian optimization to estimate the functional relationship between the hyperparameters and the RMSE. The gaussian process model attempts to discover a function that relates values of the hyperparameters to an output, in this case, the RMSE. Then an optimization function is used to seek for the combination of parameters where the function is maximized. Theoretically this should result in a set of optimal hyperparameters for a given model. Below is an example using a support vector machine (SVM) model. There are only two hyperparameters for a SVM model, the effect that the large residuals have on the regression is controlled by the *cost (C)* and the standard deviation of the Gaussian kernel is controlled by *sigma*. We don't know what values to select apriori, so we use the steps above. I have automated this process using a custom function that is a wrapper around both the `train` function from the caret package and the `BayesianOptimization` function from the rBayesianOptimization package.

```
library(caret)
source('scripts/bayes_optim_caret.R')

# list of parameter boundaries
svmg_bounds <-  list(C = c(1,5),
                     sigma = c(0,1))

# pass to custom function
svmg_params <- bayes_optim_caret(model_data,'svmRadial',svmg_bounds,iter=10,acq='ucb')
```

We can easily plot the parameter search for the SVM as it only has two parameters. The algorithm first seeds the search space with 10 random RMSE values resulting from different combinations of parameters. These are represented as the colored points below. The value is the negative RMSE in log space. We take the negation of the RMSE because the algorithm seeks to maximize the function. The line segments and the arrows represent the iterations of the Gaussian process model, with the color of the line indicating the resulting -RMSE for each iteration.



Description of train_bl_models()

We classified multivariate regression and geostatistical models as "baseline" methods. This classification scheme is used as a way to compare groups of models and does not reflect the complexity, accuracy, or robustness of the method. The models include two versions of censored regression, ordinary kriging, and a unit-area discharge null model. The first thing to note is that the response varaible is slightly different for the censored regression models.

```
model_data <- expVars$cleanBCs %>% # start with basin chars
  setNames(tolower(names(.))) %>% # make sure colname are lower case
  mutate(class = as.integer(as.factor(class))) %>% # change class to integer
  mutate(drain_sqkm = log(drain_sqkm)) %>%
  mutate_each(funs(as.numeric)) %>% # make everything numeric
  scale() %>% # convert to z-score: (x-mu)/sigma
  as.data.frame() %>% # make data frame
  mutate(y=data_full$y7q10+0.001) %>%
  mutate(y=log(y)) %>%
  select(y, class:aspect_eastness)
```

If you compare this setup to the `model_data` setup above, you will notice that the response variable is slightly different. This is because the censored regression functions require a unique censoring value. In raw 7Q10 space, this is normally just zero. If we add a small constant to the 7Q10's and normalize them by their drainage basins, then there is no longer a unique censoring value (because the drainage basin areas are different). Therefore, we just take the natural log of the 7Q10 + 0.001 and censor as ln(0.001).

The only two models that require tuning are the two versions of the censored regression models. For example, the full censored regression model requires selecting a subset of predictors using best subset selection. It is impossible to know which predictors or the best selection algorithm to select apriori, so this is tuned using cross validation. The first step is to create a grid of possible values:

```
# tune parameter grid for lcr_tobit
  lcr_grid <- expand.grid(subset_method=c("forward","backward"),
                          subset_number=c(2,4,8,10,12))
```

```
print(lcr_grid)
```

```
##    subset_method subset_number
## 1        forward             2
## 2       backward             2
## 3        forward             4
## 4       backward             4
## 5        forward             8
## 6       backward             8
## 7        forward            10
## 8       backward            10
## 9        forward            12
## 10      backward            12
```

and in this case, 10 different models are built inside a loo-cv framework and the RMSE is recorded for each model. The model with the lowest RMSE is selected as the final model. This process is referred to a "grid search".

```
  lcr_tune_error <- numeric()
  for (i in 1:nrow(lcr_grid)){

    # fit model with values from grid
    fit <- lcr.tobit(model_data,
                     subset_method=as.character(lcr_grid[i,1]),
                     subset_number=lcr_grid[i,2],
                     thold=log(0.001))

    # record rmse for ith model
    lcr_tune_error[i] <- sqrt(mean((fit$obs-fit$pred)^2,na.rm=T))
  }
```

A similar process is done for the region of influenced censored regression model, but with an additional "neighbors" hyperparameter.

```
# tune parameter grid for ROI lcr_tobit
  roi_grid <- expand.grid(neighbors=seq(25,200,25),
                          subset_method=c("forward","backward"),
                          subset_number=c(2,4,8,10,12))

  roi_tune_error <- numeric()
  for (i in 1:nrow(roi_grid)){

    # fit model with values from grid
    fit <- roi.lcr.tobit(model_data,
                         neighbors=roi_grid[i,1],
                         subset_method=as.character(roi_grid[i,2]),
                         subset_number=roi_grid[i,3],
                         thold=log(0.001))

    # record rmse for ith model
    roi_tune_error[i] <- sqrt(mean((fit$obs-fit$pred)^2,na.rm=T))
  }
```

R package citations

```
pkg_list <- c("dplyr","reshape2","randomForest","kknn","caret",
              "gbm","kernlab","Cubist","glmnet","devtools","AER",
              "leaps","doMC","ICEbox","geoR","smwrQW","smwrStats")

pkg_list <- pkg_list[order(pkg_list)]

sapply(pkg_list,function(x) print(citation(x)))
```


- Christian Kleiber and Achim Zeileis (2008). Applied Econometrics with R. New York: Springer-Verlag. ISBN 978-0-387-77316-2. URL https://CRAN.R-project.org/package=AER
- Max Kuhn. Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, Brenton Kenkel, the R Core Team, Michael Benesty, Reynald Lescarbeau, Andrew Ziem, Luca Scrucca, Yuan Tang, Can Candan and Tyler Hunt. (2017). caret: Classification and Regression Training. R package version 6.0-76. https://CRAN.R-project.org/package=caret
- Max Kuhn, Steve Weston, Chris Keefer and Nathan Coulter. C code for Cubist by Ross Quinlan (2016). Cubist: Rule- And Instance-Based Regression Modeling. R package version 0.0.19. https://CRAN.R-project.org/package=Cubist
- Hadley Wickham and Winston Chang (2017). devtools: Tools to Make Developing R Packages Easier. R package version 1.13.2. https://CRAN.R-project.org/package=devtools
- Revolution Analytics and Steve Weston (2015). doMC: Foreach Parallel Adaptor for 'parallel'. R package version 1.3.4. https://CRAN.R-project.org/package=doMC
- Hadley Wickham and Romain Francois (2016). dplyr: A Grammar of Data Manipulation. R package version 0.5.0. https://CRAN.R-project.org/package=dplyr
- Greg Ridgeway with contributions from others (2017). gbm: Generalized Boosted Regression Models. R package version 2.1.3. https://CRAN.R-project.org/package=gbm
- Paulo J. Ribeiro Jr and Peter J. Diggle (2016). geoR: Analysis of Geostatistical Data. R package version 1.7-5.2. https://CRAN.R-project.org/package=geoR
- Jerome Friedman, Trevor Hastie, Robert Tibshirani (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, 33(1), 1-22. URL http://www.jstatsoft.org/v33/i01/.
- Noah Simon, Jerome Friedman, Trevor Hastie, Rob Tibshirani (2011). Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent. Journal of Statistical Software, 39(5), 1-13. URL http://www.jstatsoft.org/v39/i05/.
- Goldstein, A., Kapelner, A., Bleich, J., and Pitkin, E., Peeking Inside the Black Box: Visualizing Statistical Learning With Plots of Individual Conditional Expectation. (2015) Journal of Computational and Graphical Statistics, 24(1): 44-65
- Alexandros Karatzoglou, Alex Smola, Kurt Hornik, Achim Zeileis (2004). kernlab - An S4 Package for Kernel Methods in R. Journal of Statistical Software 11(9), 1-20. URL http://www.jstatsoft.org/v11/i09/
- Klaus Schliep and Klaus Hechenbichler (2016). kknn: Weighted k-Nearest Neighbors. R package version 1.3.1. https://CRAN.R-project.org/package=kknn
- Thomas Lumley based on Fortran code by Alan Miller (2017). leaps: Regression Subset Selection. R package version 3.0. https://CRAN.R-project.org/package=leaps
- A. Liaw and M. Wiener (2002). Classification and Regression by randomForest. R News 2(3), 18–22.

- Hadley Wickham (2007). Reshaping Data with the reshape Package. Journal of Statistical Software, 21(12), 1-20. URL http://www.jstatsoft.org/v21/i12/.
- Lorenz, D.L., in preparation smwrQW–An R Package for Managing and Analyzing Water-Quality Data, Version 0.7.9:
- Lorenz, D.L., in preparation, smwrStats–An R package for the analysis of hydrologic data, Version 0.7.5: U.S. Geological Survey Open File Report