library(outbreaks)

library(tidyverse)

library(plyr)

library(mice)

library(caret)

library(purrr)

**The data**

The dataset contains case ID, date of onset, date of hospitalization, date of outcome, gender, age, province and of course outcome: Death or Recovery.

**Pre-processing**

**Change: variable names (i.e. column names) have been renamed, dots have been replaced with underscores, letters are all lower case now.**

**Change: I am using the tidyverse notation more consistently.**

First, I'm doing some preprocessing, including:

* renaming missing data as NA
* adding an ID column
* setting column types
* gathering date columns
* changing factor names of dates (to make them look nicer in plots) and of province (to combine provinces with few cases)

fluH7N9\_china\_2013$age[which(fluH7N9\_china\_2013$age == "?")] <- NA

fluH7N9\_china\_2013\_gather <- fluH7N9\_china\_2013 %>%

mutate(case\_id = paste("case", case\_id, sep = "\_"),

age = as.numeric(age)) %>%

gather(Group, Date, date\_of\_onset:date\_of\_outcome) %>%

mutate(Group = as.factor(mapvalues(Group, from = c("date\_of\_onset", "date\_of\_hospitalisation", "date\_of\_outcome"),

to = c("date of onset", "date of hospitalisation", "date of outcome"))),

province = mapvalues(province, from = c("Anhui", "Beijing", "Fujian", "Guangdong", "Hebei", "Henan", "Hunan", "Jiangxi", "Shandong", "Taiwan"), to = rep("Other", 10)))

I'm also

* adding a third gender level for unknown gender

levels(fluH7N9\_china\_2013\_gather$gender) <- c(levels(fluH7N9\_china\_2013\_gather$gender), "unknown")

fluH7N9\_china\_2013\_gather$gender[is.na(fluH7N9\_china\_2013\_gather$gender)] <- "unknown"

head(fluH7N9\_china\_2013\_gather)

*## case\_id outcome gender age province Group Date*

*## 1 case\_1 Death m 58 Shanghai date of onset 2013-02-19*

*## 2 case\_2 Death m 7 Shanghai date of onset 2013-02-27*

*## 3 case\_3 Death f 11 Other date of onset 2013-03-09*

*## 4 case\_4 <NA> f 18 Jiangsu date of onset 2013-03-19*

*## 5 case\_5 Recover f 20 Jiangsu date of onset 2013-03-19*

*## 6 case\_6 Death f 9 Jiangsu date of onset 2013-03-21*

For plotting, I am defining a custom ggplot2 theme:

my\_theme <- function(base\_size = 12, base\_family = "sans"){

theme\_minimal(base\_size = base\_size, base\_family = base\_family) +

theme(

axis.text = element\_text(size = 12),

axis.text.x = element\_text(angle = 45, vjust = 0.5, hjust = 0.5),

axis.title = element\_text(size = 14),

panel.grid.major = element\_line(color = "grey"),

panel.grid.minor = element\_blank(),

panel.background = element\_rect(fill = "aliceblue"),

strip.background = element\_rect(fill = "lightgrey", color = "grey", size = 1),

strip.text = element\_text(face = "bold", size = 12, color = "black"),

legend.position = "bottom",

legend.justification = "top",

legend.box = "horizontal",

legend.box.background = element\_rect(colour = "grey50"),

legend.background = element\_blank(),

panel.border = element\_rect(color = "grey", fill = NA, size = 0.5)

)

}

And use that theme to visualize the data:

ggplot(data = fluH7N9\_china\_2013\_gather, aes(x = Date, y = age, fill = outcome)) +

stat\_density2d(aes(alpha = ..level..), geom = "polygon") +

geom\_jitter(aes(color = outcome, shape = gender), size = 1.5) +

geom\_rug(aes(color = outcome)) +

scale\_y\_continuous(limits = c(0, 90)) +

labs(

fill = "Outcome",

color = "Outcome",

alpha = "Level",

shape = "Gender",

x = "Date in 2013",

y = "Age",

title = "2013 Influenza A H7N9 cases in China",

subtitle = "Dataset from 'outbreaks' package (Kucharski et al. 2014)",

caption = ""

) +

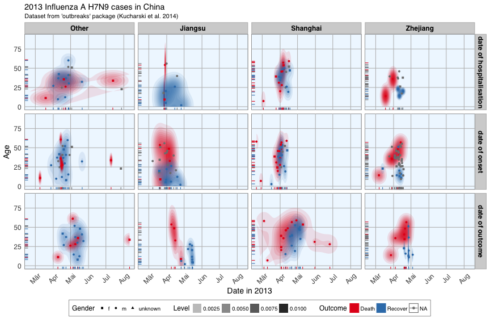
facet\_grid(Group ~ province) +

my\_theme() +

scale\_shape\_manual(values = c(15, 16, 17)) +

scale\_color\_brewer(palette="Set1", na.value = "grey50") +

scale\_fill\_brewer(palette="Set1")

Gives this plot:  
[](https://datascienceplus.com/wp-content/uploads/2018/05/main-1.png)

ggplot(data = fluH7N9\_china\_2013\_gather, aes(x = Date, y = age, color = outcome)) +

geom\_point(aes(color = outcome, shape = gender), size = 1.5, alpha = 0.6) +

geom\_path(aes(group = case\_id)) +

facet\_wrap( ~ province, ncol = 2) +

my\_theme() +

scale\_shape\_manual(values = c(15, 16, 17)) +

scale\_color\_brewer(palette="Set1", na.value = "grey50") +

scale\_fill\_brewer(palette="Set1") +

labs(

color = "Outcome",

shape = "Gender",

x = "Date in 2013",

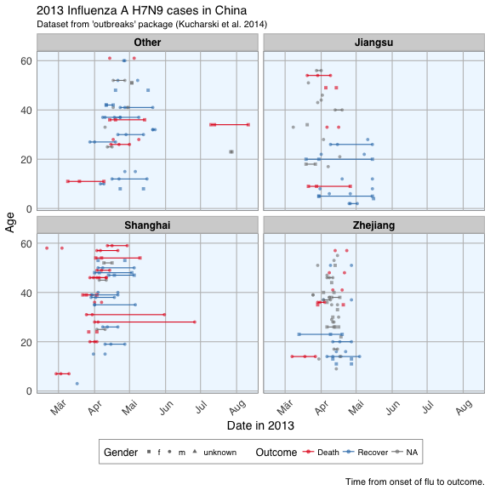
y = "Age",

title = "2013 Influenza A H7N9 cases in China",

subtitle = "Dataset from 'outbreaks' package (Kucharski et al. 2014)",

caption = "\nTime from onset of flu to outcome."

)

Gives this plot:  
[](https://datascienceplus.com/wp-content/uploads/2018/05/unnamed-chunk-6-1.png)

**Features**

In machine learning-speak features are what we call the variables used for model training. Using the right features dramatically influences the accuracy and success of your model. For this example, I am keeping age, but I am also generating new features from the date information and converting gender and province into numerical values.

dataset <- fluH7N9\_china\_2013 %>%

mutate(hospital = as.factor(ifelse(is.na(date\_of\_hospitalisation), 0, 1)),

gender\_f = as.factor(ifelse(gender == "f", 1, 0)),

province\_Jiangsu = as.factor(ifelse(province == "Jiangsu", 1, 0)),

province\_Shanghai = as.factor(ifelse(province == "Shanghai", 1, 0)),

province\_Zhejiang = as.factor(ifelse(province == "Zhejiang", 1, 0)),

province\_other = as.factor(ifelse(province == "Zhejiang" | province == "Jiangsu" | province == "Shanghai", 0, 1)),

days\_onset\_to\_outcome = as.numeric(as.character(gsub(" days", "",

as.Date(as.character(date\_of\_outcome), format = "%Y-%m-%d") -

as.Date(as.character(date\_of\_onset), format = "%Y-%m-%d")))),

days\_onset\_to\_hospital = as.numeric(as.character(gsub(" days", "",

as.Date(as.character(date\_of\_hospitalisation), format = "%Y-%m-%d") -

as.Date(as.character(date\_of\_onset), format = "%Y-%m-%d")))),

age = age,

early\_onset = as.factor(ifelse(date\_of\_onset < summary(fluH7N9\_china\_2013$date\_of\_onset)[[3]], 1, 0)),

early\_outcome = as.factor(ifelse(date\_of\_outcome < summary(fluH7N9\_china\_2013$date\_of\_outcome)[[3]], 1, 0))) %>%

subset(select = -c(2:4, 6, 8))

rownames(dataset) <- dataset$case\_id

dataset[, -2] <- as.numeric(as.matrix(dataset[, -2]))

head(dataset)

*## case\_id outcome age hospital gender\_f province\_Jiangsu province\_Shanghai*

*## 1 1 Death 87 0 0 0 1*

*## 2 2 Death 27 1 0 0 1*

*## 3 3 Death 35 1 1 0 0*

*## 4 4 <NA> 45 1 1 1 0*

*## 5 5 Recover 48 1 1 1 0*

*## 6 6 Death 32 1 1 1 0*

*## province\_Zhejiang province\_other days\_onset\_to\_outcome*

*## 1 0 0 13*

*## 2 0 0 11*

*## 3 0 1 31*

*## 4 0 0 NA*

*## 5 0 0 57*

*## 6 0 0 36*

*## days\_onset\_to\_hospital early\_onset early\_outcome*

*## 1 NA 1 1*

*## 2 4 1 1*

*## 3 10 1 1*

*## 4 8 1 NA*

*## 5 11 1 0*

*## 6 7 1 1*

summary(dataset$outcome)

*## Death Recover NA's*

*## 32 47 57*

**Imputing missing values**

I am using the [mice package for imputing missing values](https://gerkovink.github.io/miceVignettes/Ad_hoc_and_mice/Ad_hoc_methods.html). DataScience+ has also other tutorials [how to impute data with MICE](https://datascienceplus.com/tag/mice-package/).

**Note:** Since publishing this blog post I learned that the idea behind using mice is to compare different imputations to see how stable they are, instead of picking one imputed set as fixed for the remainder of the analysis. Therefore, I changed the focus of this post a little bit: in the old post I compared many different algorithms and their outcome; in this updated version I am only showing the Random Forest algorithm and focus on comparing the different imputed datasets. I am ignoring feature importance and feature plots because nothing changed compared to the old post.

* md.pattern() shows the pattern of missingness in the data:

md.pattern(dataset)

*## case\_id hospital province\_Jiangsu province\_Shanghai province\_Zhejiang*

*## 42 1 1 1 1 1*

*## 27 1 1 1 1 1*

*## 2 1 1 1 1 1*

*## 2 1 1 1 1 1*

*## 18 1 1 1 1 1*

*## 1 1 1 1 1 1*

*## 36 1 1 1 1 1*

*## 3 1 1 1 1 1*

*## 3 1 1 1 1 1*

*## 2 1 1 1 1 1*

*## 0 0 0 0 0*

*## province\_other age gender\_f early\_onset outcome early\_outcome*

*## 42 1 1 1 1 1 1*

*## 27 1 1 1 1 1 1*

*## 2 1 1 1 1 1 0*

*## 2 1 1 1 0 1 1*

*## 18 1 1 1 1 0 0*

*## 1 1 1 1 1 1 0*

*## 36 1 1 1 1 0 0*

*## 3 1 1 1 0 1 0*

*## 3 1 1 1 0 0 0*

*## 2 1 0 0 0 1 0*

*## 0 2 2 10 57 65*

*## days\_onset\_to\_outcome days\_onset\_to\_hospital*

*## 42 1 1 0*

*## 27 1 0 1*

*## 2 0 1 2*

*## 2 0 0 3*

*## 18 0 1 3*

*## 1 0 0 3*

*## 36 0 0 4*

*## 3 0 0 4*

*## 3 0 0 5*

*## 2 0 0 6*

*## 67 74 277*

* mice() generates the imputations

dataset\_impute <- mice(data = dataset[, -2], print = FALSE)

* by default, mice() calculates five (m = 5) imputed data sets
* we can combine them all in one output with the complete("long") function
* I did not want to impute missing values in the outcome column, so I have to merge it back in with the imputed data

datasets\_complete <- right\_join(dataset[, c(1, 2)],

complete(dataset\_impute, "long"),

by = "case\_id") %>%

select(-.id)

head(datasets\_complete)

*## case\_id outcome .imp age hospital gender\_f province\_Jiangsu*

*## 1 1 Death 1 87 0 0 0*

*## 2 2 Death 1 27 1 0 0*

*## 3 3 Death 1 35 1 1 0*

*## 4 4 <NA> 1 45 1 1 1*

*## 5 5 Recover 1 48 1 1 1*

*## 6 6 Death 1 32 1 1 1*

*## province\_Shanghai province\_Zhejiang province\_other days\_onset\_to\_outcome*

*## 1 1 0 0 13*

*## 2 1 0 0 11*

*## 3 0 0 1 31*

*## 4 0 0 0 20*

*## 5 0 0 0 57*

*## 6 0 0 0 36*

*## days\_onset\_to\_hospital early\_onset early\_outcome*

*## 1 5 1 1*

*## 2 4 1 1*

*## 3 10 1 1*

*## 4 8 1 1*

*## 5 11 1 0*

*## 6 7 1 1*

Let's compare the distributions of the five different imputed datasets:

datasets\_complete %>%

gather(x, y, age:early\_outcome) %>%

ggplot(aes(x = y, fill = .imp, color = .imp)) +

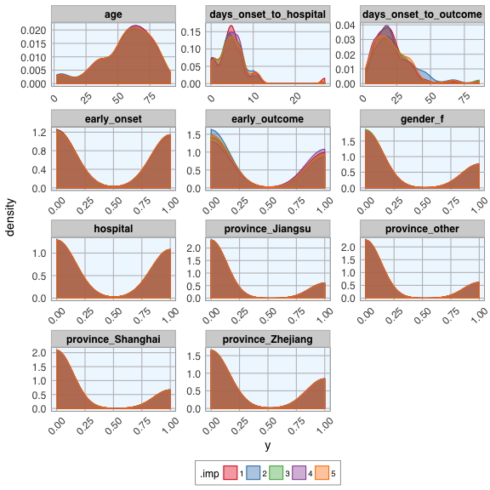
facet\_wrap(~ x, ncol = 3, scales = "free") +

geom\_density(alpha = 0.4) +

scale\_fill\_brewer(palette="Set1", na.value = "grey50") +

scale\_color\_brewer(palette="Set1", na.value = "grey50") +

my\_theme()

Gives this plot:  
[](https://datascienceplus.com/wp-content/uploads/2018/05/unnamed-chunk-13-1.png)

**Test, train and validation data sets**

Now, we can go ahead with machine learning!

The dataset contains a few missing values in the outcome column; those will be the test set used for final predictions (see the old blog post for this).

train\_index <- which(is.na(datasets\_complete$outcome))

train\_data <- datasets\_complete[-train\_index, ]

test\_data <- datasets\_complete[train\_index, -2]

The remainder of the data will be used for modeling. Here, I am splitting the data into 70% training and 30% test data.

Because I want to model each imputed dataset separately, I am using the nest() and map() functions.

set.seed(42)

val\_data <- train\_data %>%

group\_by(.imp) %>%

nest() %>%

mutate(val\_index = map(data, ~ createDataPartition(.$outcome, p = 0.7, list = FALSE)),

val\_train\_data = map2(data, val\_index, ~ .x[.y, ]),

val\_test\_data = map2(data, val\_index, ~ .x[-.y, ]))

**Machine Learning algorithms**

**Random Forest**

To make the code tidier, I am first defining the modeling function with the parameters I want.

model\_function <- function(df) {

caret::train(outcome ~ .,

data = df,

method = "rf",

preProcess = c("scale", "center"),

trControl = trainControl(method = "repeatedcv", number = 5, repeats = 3, verboseIter = FALSE))

}

Next, I am using the nested tibble from before to map() the model function, predict the outcome and calculate confusion matrices.

set.seed(42)

val\_data\_model <- val\_data %>%

mutate(model = map(val\_train\_data, ~ model\_function(.x)),

predict = map2(model, val\_test\_data, ~ data.frame(prediction = predict(.x, .y[, -2]))),

predict\_prob = map2(model, val\_test\_data, ~ data.frame(outcome = .y[, 2],

prediction = predict(.x, .y[, -2], type = "prob"))),

confusion\_matrix = map2(val\_test\_data, predict, ~ confusionMatrix(.x$outcome, .y$prediction)),

confusion\_matrix\_tbl = map(confusion\_matrix, ~ as.tibble(.x$table)))

**Comparing accuracy of models**

To compare how the different imputations did, I am plotting

* the confusion matrices:

val\_data\_model %>%

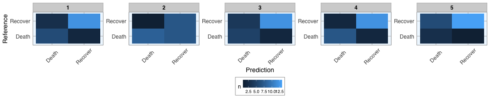
unnest(confusion\_matrix\_tbl) %>%

ggplot(aes(x = Prediction, y = Reference, fill = n)) +

facet\_wrap(~ .imp, ncol = 5, scales = "free") +

geom\_tile() +

my\_theme()

Gives this plot:  
[](https://datascienceplus.com/wp-content/uploads/2018/05/unnamed-chunk-18-1.png)

* and the prediction probabilities for correct and wrong predictions:

val\_data\_model %>%

unnest(predict\_prob) %>%

gather(x, y, prediction.Death:prediction.Recover) %>%

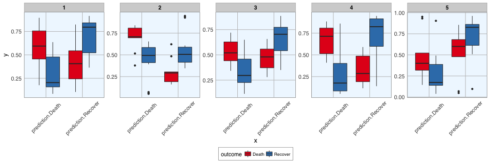
ggplot(aes(x = x, y = y, fill = outcome)) +

facet\_wrap(~ .imp, ncol = 5, scales = "free") +

geom\_boxplot() +

scale\_fill\_brewer(palette="Set1", na.value = "grey50") +

my\_theme()

Gives this plot:  
[](https://datascienceplus.com/wp-content/uploads/2018/05/unnamed-chunk-19-1.png)

Hope, you found that example interesting and helpful!

sessionInfo()

*## R version 3.5.0 (2018-04-23)*

*## Platform: x86\_64-apple-darwin15.6.0 (64-bit)*

*## Running under: macOS High Sierra 10.13.4*

*##*

*## Matrix products: default*

*## BLAS: /System/Library/Frameworks/Accelerate.framework/Versions/A/Frameworks/vecLib.framework/Versions/A/libBLAS.dylib*

*## LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib*

*##*

*## locale:*

*## [1] de\_DE.UTF-8/de\_DE.UTF-8/de\_DE.UTF-8/C/de\_DE.UTF-8/de\_DE.UTF-8*

*##*

*## attached base packages:*

*## [1] stats graphics grDevices utils datasets methods base*

*##*

*## other attached packages:*

*## [1] bindrcpp\_0.2.2 knitr\_1.20 RWordPress\_0.2-3 caret\_6.0-79*

*## [5] mice\_2.46.0 lattice\_0.20-35 plyr\_1.8.4 forcats\_0.3.0*

*## [9] stringr\_1.3.1 dplyr\_0.7.4 purrr\_0.2.4 readr\_1.1.1*

*## [13] tidyr\_0.8.0 tibble\_1.4.2 ggplot2\_2.2.1 tidyverse\_1.2.1*

*## [17] outbreaks\_1.3.0*

*##*

*## loaded via a namespace (and not attached):*

*## [1] nlme\_3.1-137 bitops\_1.0-6 lubridate\_1.7.4*

*## [4] RColorBrewer\_1.1-2 dimRed\_0.1.0 httr\_1.3.1*

*## [7] rprojroot\_1.3-2 tools\_3.5.0 backports\_1.1.2*

*## [10] R6\_2.2.2 rpart\_4.1-13 lazyeval\_0.2.1*

*## [13] colorspace\_1.3-2 nnet\_7.3-12 withr\_2.1.2*

*## [16] tidyselect\_0.2.4 mnormt\_1.5-5 compiler\_3.5.0*

*## [19] cli\_1.0.0 rvest\_0.3.2 xml2\_1.2.0*

*## [22] labeling\_0.3 bookdown\_0.7 scales\_0.5.0*

*## [25] sfsmisc\_1.1-2 DEoptimR\_1.0-8 psych\_1.8.4*

*## [28] robustbase\_0.93-0 randomForest\_4.6-14 digest\_0.6.15*

*## [31] foreign\_0.8-70 rmarkdown\_1.9 pkgconfig\_2.0.1*

*## [34] htmltools\_0.3.6 highr\_0.6 rlang\_0.2.0*

*## [37] readxl\_1.1.0 ddalpha\_1.3.3 rstudioapi\_0.7*

*## [40] XMLRPC\_0.3-0 bindr\_0.1.1 jsonlite\_1.5*

*## [43] ModelMetrics\_1.1.0 RCurl\_1.95-4.10 magrittr\_1.5*

*## [46] Matrix\_1.2-14 Rcpp\_0.12.16 munsell\_0.4.3*

*## [49] abind\_1.4-5 stringi\_1.2.2 yaml\_2.1.19*

*## [52] MASS\_7.3-50 recipes\_0.1.2 grid\_3.5.0*

*## [55] parallel\_3.5.0 crayon\_1.3.4 haven\_1.1.1*

*## [58] splines\_3.5.0 hms\_0.4.2 pillar\_1.2.2*

*## [61] reshape2\_1.4.3 codetools\_0.2-15 stats4\_3.5.0*

*## [64] CVST\_0.2-1 magic\_1.5-8 XML\_3.98-1.11*

*## [67] glue\_1.2.0 evaluate\_0.10.1 blogdown\_0.6*

*## [70] modelr\_0.1.2 foreach\_1.4.4 cellranger\_1.1.0*

*## [73] gtable\_0.2.0 kernlab\_0.9-26 assertthat\_0.2.0*

*## [76] DRR\_0.0.3 xfun\_0.1 gower\_0.1.2*

*## [79] prodlim\_2018.04.18 broom\_0.4.4 e1071\_1.6-8*

*## [82] class\_7.3-14 survival\_2.42-3 geometry\_0.3-6*

*## [85] timeDate\_3043.102 RcppRoll\_0.2.2 iterators\_1.0.9*

*## [88] lava\_1.6.1 ipred\_0.9-6*