Pritchard2014_Graph_Generation

Contents

Introduction

This document describes the production of the Figure 1 graphs from the Pritchard and Birch (2014) opinion paper in *Mol. Plant. Pathol.* (doi:10.1111/mpp.12210). In that manuscript, a toy dynamic model of the plant immune system is presented (this is deposited in the BioModels database with accession MODEL1408280000: http://www.ebi.ac.uk/biomodels-main/). This document uses the output from five runs of the model using COPASI, representing 200s timecourses:

- `no_path.txt`: The default model, with `k1("Pathogen arrival")=0`
- `no_pti_no_supp_no_eti.txt`: Default model, with `k1("PAMP recognition")=0`, `k1("Callose suppression")=0`, and `k1("Effector recognition")=0`
- `pti_no_supp_no_eti.txt`: Default model, with `k1("Callose suppression")=0`, and `k1("Effector recognition")=0`
- `pti_supp_no_eti.txt`: Default model, with `k1("Effector recognition")=0`
- `pti_supp_eti.txt`: Default model

Load model output into dataframes

COPASI tab-separated output starts the header line with the # character, which is annoyingly also the comment line character in R, so we need to specify that no comment characters should be respected, when we load the dataframes. We initially store each run in its own dataframe, then bind the output into a single dataframe, with columns `Time`, `variable`, `value`, and `runtype` where `Time` is the model run timepoint, `variable` is either `Callose` or `Pathogen` (the two variables of interest), `value` is the dimensionless level of the variable, and `runtype` is a human-readable label for the run.

```
# We use ggplot2, and the handy reshape2 libraries
library(ggplot2)
library(reshape2)

# Load data
# These files are timecourse output from COPASI, but COPASI's header line
# starts with the default comment character, so we need to lose this.
# (possibly by manual editing)
```

```
nopathdata = read.table('no_path.txt', sep='\t',
                        header=T, comment.char="")
pathdata = read.table('no_pti_no_supp_no_eti.txt', sep='\t',
                      header=T, comment.char="")
ptidata = read.table('pti_no_supp_no_eti.txt', sep='\t',
                     header=T, comment.char="")
ptietsdata = read.table('pti_supp_no_eti.txt', sep='\t',
                        header=T, comment.char="")
ptietsetidata = read.table('pti_supp_eti.txt', sep='\t',
                           header=T, comment.char="")

# Reshape all five datasets and lose the columns we're not interested in
dflist = list(nopath=nopathdata, noresponse=pathdata,
              pti=ptidata, ptiets=ptietsdata,
              ptietseti=ptietsetidata)
dflist = lapply(dflist, function(df) {
  colnames(df)[1] = "Time"
  colnames(df)[7] = "Pathogen"
  df = melt(df, id=c("Time"))
  df = df[df$variable %in% c("Callose", "Pathogen"),]
})

# Bind all our data into a single dataframe
data = do.call(rbind, dflist)
# Convert row.names to run names
vals = 402  # How many rows per run?
data$runtype = c(rep("No Pathogen", vals), rep("No Response", vals),
                 rep("PTI", vals), rep("PTI+ETS", vals),
                 rep("PTI+ETS+ETI", vals))
```

Plot steady-states

The code below renders a plot of steady-state values of `Pathogen` and `Callose` variables across the five run types. The output behaves as we might expect.

```
# We assume that the model reaches a steady-state (or near enough), by
# Time=200s, and just pull out those values
ssdata = data[data$Time==200,]
print(ssdata)
```

```
##                 Time variable    value     runtype
## nopath.1005      200  Callose 0.000000 No Pathogen
## nopath.1206      200 Pathogen 0.000000 No Pathogen
## noresponse.1005  200  Callose 0.000000 No Response
## noresponse.1206  200 Pathogen 1.000000 No Response
## pti.1005         200  Callose 0.414207         PTI
## pti.1206         200 Pathogen 0.707112         PTI
## ptiets.1005      200  Callose 0.300568     PTI+ETS
## ptiets.1206      200 Pathogen 0.768893     PTI+ETS
## ptietseti.1005   200  Callose 0.280057 PTI+ETS+ETI
## ptietseti.1206   200 Pathogen 0.640030 PTI+ETS+ETI
```

When no pathogen is present, `Pathogen` and `Callose` (a proxy for overall PTI response) are both zero. If pathogen is introduced to a host that does not produce PTI or ETI (`No Response`), then `Pathogen` reaches its full arbitrary level of 1.

However, if the host exhibits PTI, and the pathogen does not produce effectors, the result is that `Callose` is produced (0.41), and the incidence of `Pathogen` falls (0.71). Against this background, if the pathogen is capable of producing effectors that result in callose supression, then steady-state `Callose` falls (0.30), and `Pathogen` rises (0.76).

Finally, for a model where the host exhibits both PTI and ETI, but the pathogen produces an effector that can suppress callose, both `Callose` and `Pathogen` fall to their lowest steady-state values (0.28 and 0.64, respectively).

```
# Plot data as a line graph of the final (steady-state) levels of
# Callose and Pathogen in each system
p1 <- ggplot(ssdata, aes(x=runtype, y=value, group=variable, colour=variable))
p1 = p1 + geom_line(size=1) +
  labs(title="Pathogen, Callose, by host type",
       x="Host response type", y="Arbitrary units")
# Save to PDF
ggsave("Figure_1c.pdf", width=8, height=8)
```

Plot time series

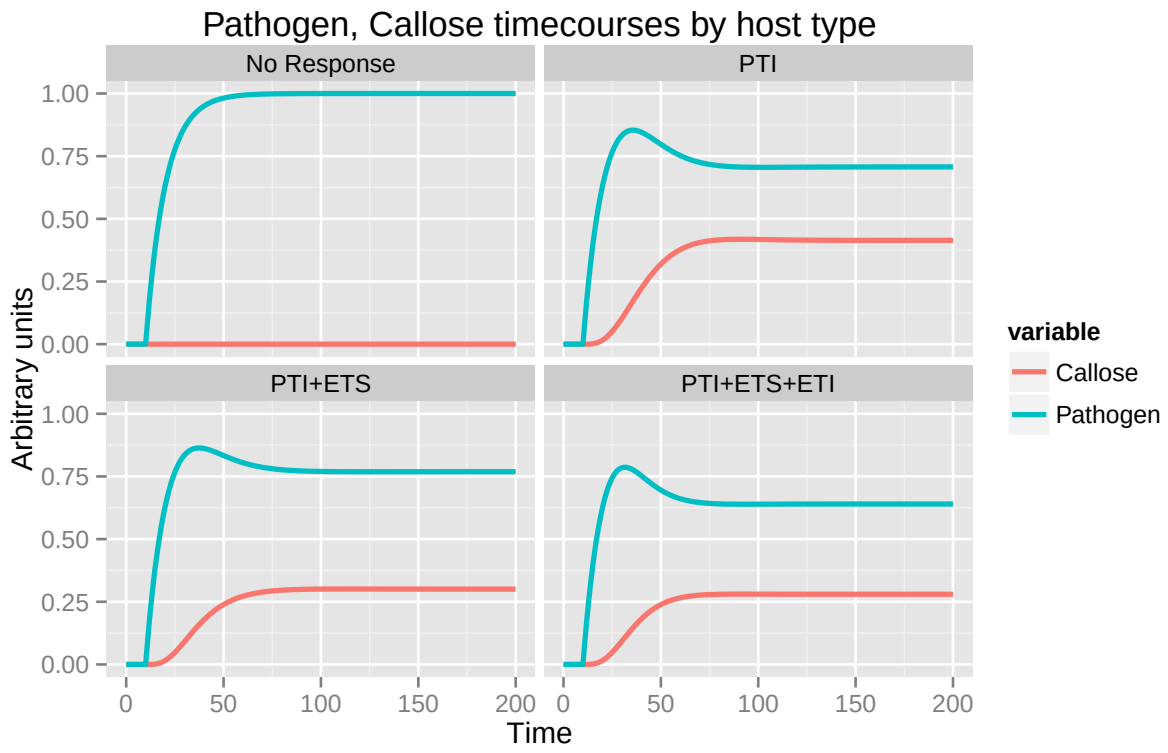
The code below renders a small multiple/facet plot for the four timecourses, where a pathogen is present. These are presented as plots of the values for `Path` (local microbial population) and `Callose` (abstracted callose deposition/PTI level), as a function of time. For these plots, we exclude the run with no pathogen present.

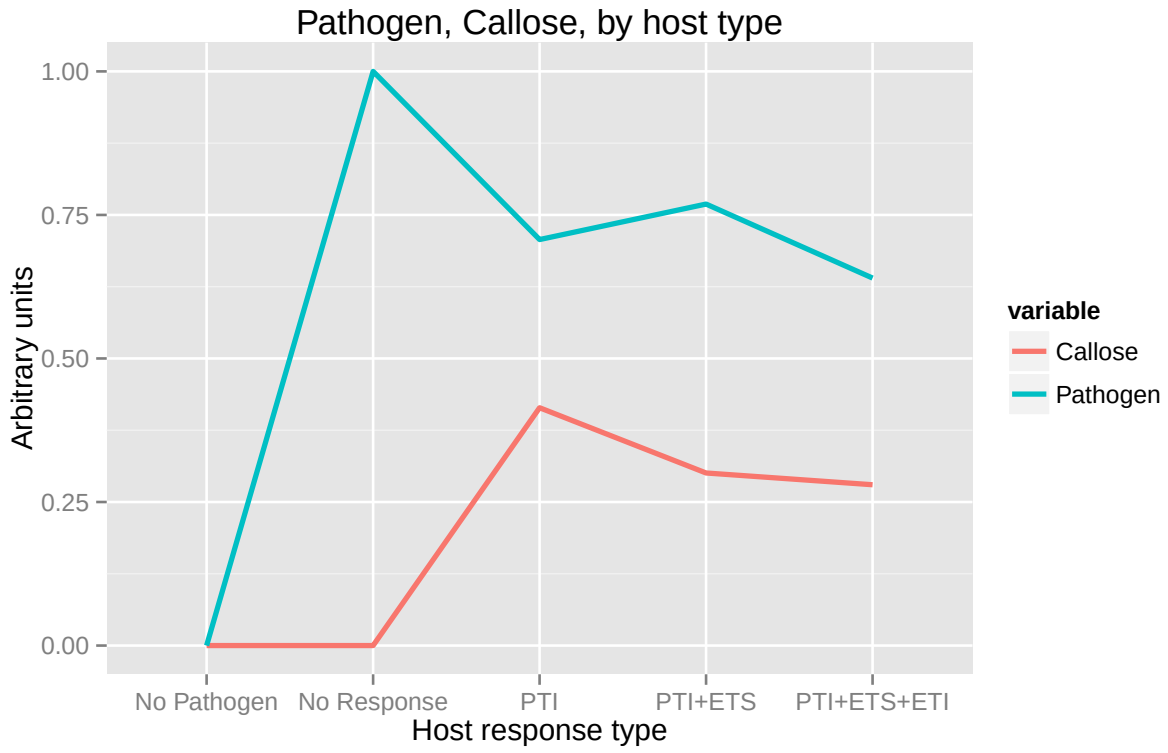
```
# We don't need to consider the run where no pathogen was present
timedata = data[!(data$runtype=="No Pathogen"),]

# Plot data as a timecourse of Callose and Pathogen for each system
p2 <- ggplot(timedata, aes(x=Time, y=value, group=variable, colour=variable))
p2 = p2 + geom_line(size=1) + labs(title="Pathogen, Callose timecourses by host type",
                                   x="Time", y="Arbitrary units") +
  facet_wrap(~ runtype, ncol=2)
# Save to PDF
ggsave("Figure_1b.pdf", width=8, height=8)
```

Output

The two figures are plotted below.

```
library(gridExtra)
grid.arrange(p2, p1, ncol=1)
```


Pathogen, Callose timecourses by host type
No Response
PTI
PTI+ETS
PTI+ETS+ETI
Arbitrary units
1.00
0.75
0.50
0.25
0.00
0
50
100
150
200
Time
variable
Callose
Pathogen


Pathogen, Callose, by host type
Arbitrary units
1.00
0.75
0.50
0.25
0.00
No Pathogen
No Response
PTI
PTI+ETS
PTI+ETS+ETI
Host response type
variable
Callose
Pathogen