An introduction to *outbreaker* 1.1-0

Thibaut Jombart

December 13, 2013

Abstract

This vignette introduces the main functionalities of *outbreaker*, a package implementing a model for disease outbreak reconstruction using epidemiological data and pathogen genome sequences. The emphasis of this document is put on using *outbreaker* and exploiting its results, more than providing an introduction to disease outbreak reconstruction. For this, see online tutorials available on the *R-epi project*: https://sites.google.com/site/therepiproject/tutorials.

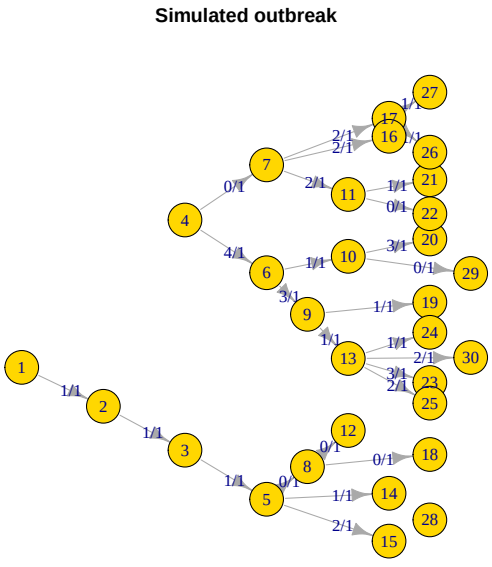
Contents

1 Running outbreaker

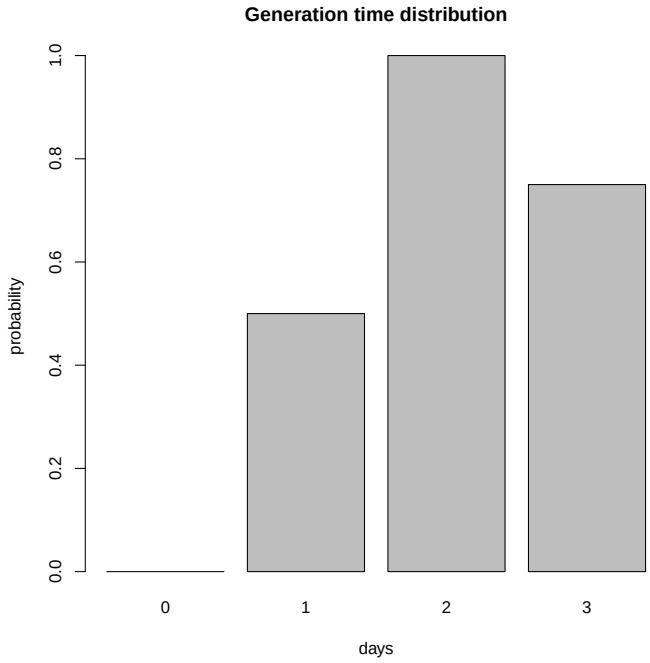
1.1 A simple example

```
library(outbreaker)
data("fakeOutbreak")
names(fakeOutbreak)

## [1] "dat"         "w"           "collecDates" "res"
```

```
dat <- fakeOutbreak$dat
w <- fakeOutbreak$w
collecDates <- fakeOutbreak$collecDates
plot(dat, main="Simulated outbreak")
```



```
barplot(w, main="Generation time distribution", ylab="probability", xlab="days", names=0:3)
```




```
res <-  outbreaker.parallel(n.runs=4, dna=dat$dna,
                            dates=collecDates,w.dens=w, n.iter=5e4)
```

```
names(res)
```

```
##  [1] "chains"         "collec.dates"   "w"              "f"
##  [5] "D"              "idx.dna"        "tune.end"       "find.import"
##  [9] "burnin"         "find.import.at" "n.runs"         "call"
```

```
names(res$chains)
```

```
##   [1] "step"     "post"     "like"     "prior"    "mu1"      "mu2"
##   [7] "gamma"    "pi"       "spa1"     "spa2"     "Tinf_1"   "Tinf_2"
##  [13] "Tinf_3"   "Tinf_4"   "Tinf_5"   "Tinf_6"   "Tinf_7"   "Tinf_8"
##  [19] "Tinf_9"   "Tinf_10"  "Tinf_11"  "Tinf_12"  "Tinf_13"  "Tinf_14"
##  [25] "Tinf_15"  "Tinf_16"  "Tinf_17"  "Tinf_18"  "Tinf_19"  "Tinf_20"
##  [31] "Tinf_21"  "Tinf_22"  "Tinf_23"  "Tinf_24"  "Tinf_25"  "Tinf_26"
##  [37] "Tinf_27"  "Tinf_28"  "Tinf_29"  "Tinf_30"  "alpha_1"  "alpha_2"
##  [43] "alpha_3"  "alpha_4"  "alpha_5"  "alpha_6"  "alpha_7"  "alpha_8"
##  [49] "alpha_9"  "alpha_10" "alpha_11" "alpha_12" "alpha_13" "alpha_14"
##  [55] "alpha_15" "alpha_16" "alpha_17" "alpha_18" "alpha_19" "alpha_20"
##  [61] "alpha_21" "alpha_22" "alpha_23" "alpha_24" "alpha_25" "alpha_26"
##  [67] "alpha_27" "alpha_28" "alpha_29" "alpha_30" "kappa_1"  "kappa_2"
##  [73] "kappa_3"  "kappa_4"  "kappa_5"  "kappa_6"  "kappa_7"  "kappa_8"
##  [79] "kappa_9"  "kappa_10" "kappa_11" "kappa_12" "kappa_13" "kappa_14"
```

```
##  [85] "kappa_15" "kappa_16" "kappa_17" "kappa_18" "kappa_19" "kappa_20"
##  [91] "kappa_21" "kappa_22" "kappa_23" "kappa_24" "kappa_25" "kappa_26"
##  [97] "kappa_27" "kappa_28" "kappa_29" "kappa_30" "run"
```

```
class(res)

## [1] "list"

names(res)

##  [1] "chains"         "collec.dates"   "w"              "f"
##  [5] "D"              "idx.dna"        "tune.end"       "find.import"
##  [9] "burnin"         "find.import.at" "n.runs"         "call"
```

The object **res** is a list with a number of named items, described in **?outbreaker**. The most important one is **res$chains**, containing the MCMC outputs:

```
class(res$chains)

## [1] "data.frame"

dim(res$chains)

## [1] 804 101

names(res$chains)

##   [1] "step"     "post"     "like"     "prior"    "mu1"      "mu2"
##   [7] "gamma"    "pi"       "spa1"     "spa2"     "Tinf_1"   "Tinf_2"
##  [13] "Tinf_3"   "Tinf_4"   "Tinf_5"   "Tinf_6"   "Tinf_7"   "Tinf_8"
##  [19] "Tinf_9"   "Tinf_10"  "Tinf_11"  "Tinf_12"  "Tinf_13"  "Tinf_14"
##  [25] "Tinf_15"  "Tinf_16"  "Tinf_17"  "Tinf_18"  "Tinf_19"  "Tinf_20"
##  [31] "Tinf_21"  "Tinf_22"  "Tinf_23"  "Tinf_24"  "Tinf_25"  "Tinf_26"
##  [37] "Tinf_27"  "Tinf_28"  "Tinf_29"  "Tinf_30"  "alpha_1"  "alpha_2"
##  [43] "alpha_3"  "alpha_4"  "alpha_5"  "alpha_6"  "alpha_7"  "alpha_8"
##  [49] "alpha_9"  "alpha_10" "alpha_11" "alpha_12" "alpha_13" "alpha_14"
##  [55] "alpha_15" "alpha_16" "alpha_17" "alpha_18" "alpha_19" "alpha_20"
##  [61] "alpha_21" "alpha_22" "alpha_23" "alpha_24" "alpha_25" "alpha_26"
##  [67] "alpha_27" "alpha_28" "alpha_29" "alpha_30" "kappa_1"  "kappa_2"
##  [73] "kappa_3"  "kappa_4"  "kappa_5"  "kappa_6"  "kappa_7"  "kappa_8"
##  [79] "kappa_9"  "kappa_10" "kappa_11" "kappa_12" "kappa_13" "kappa_14"
##  [85] "kappa_15" "kappa_16" "kappa_17" "kappa_18" "kappa_19" "kappa_20"
##  [91] "kappa_21" "kappa_22" "kappa_23" "kappa_24" "kappa_25" "kappa_26"
##  [97] "kappa_27" "kappa_28" "kappa_29" "kappa_30" "run"

res$chains[1:10,1:10]

##    step    post    like prior       mu1       mu2 gamma     pi spa1 spa2
## 1     1 -1106.3 -1108.4 2.093 5.000e-05 5.000e-05     1 0.9770    0    0
## 2   500  -468.0  -470.2 2.144 5.021e-05 5.021e-05     1 0.9826    0    0
## 3  1000  -446.9  -448.8 1.909 5.038e-05 5.038e-05     1 0.9572    0    0
## 4  1500  -446.7  -448.9 2.294 5.060e-05 5.060e-05     1 0.9990    0    0
```

```
## 5  2000  -446.5  -448.7 2.227 5.087e-05 5.087e-05     1 0.9916    0    0
## 6  2500  -446.6  -448.7 2.147 5.080e-05 5.080e-05     1 0.9829    0    0
## 7  3000  -446.9  -449.1 2.216 5.138e-05 5.138e-05     1 0.9905    0    0
## 8  3500  -448.5  -450.6 2.168 5.078e-05 5.078e-05     1 0.9852    0    0
## 9  4000  -446.8  -449.1 2.233 5.309e-05 5.309e-05     1 0.9923    0    0
## 10 4500  -452.0  -453.8 1.855 5.323e-05 5.323e-05     1 0.9515    0    0
```

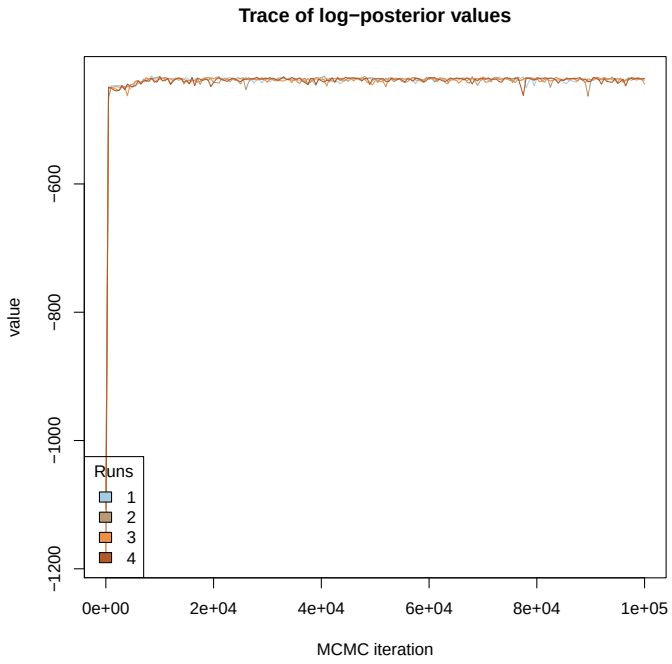
The columns of this `data.frame` store the following outputs:

- `step`: the MCMC iteration of the sample
- `post/like/prior`: log values for posterior, likelihood, and prior densities
- `mu1`: in mutation model 1, mutation rate; otherwise, the rate of transitions, per site and generation
- `mu2`: in mutation model 1, mutation rate (mu1=mu2); otherwise, the rate of transversions, per site and generation
- `gamma`: the ratio between transversions and transitions (μ_2/μ_1)
- `pi`: the proportion of the transmission tree sampled
- `Tinf_[number]`: dates of infection
- `alpha_[number]`: the index of the ancestral cases (infectors)
- `kappa_[number]`: the number of generations between cases and their most recent sampled ancestor (here, fixed to 1)
- `run`: for parallel runs, the index of the run.

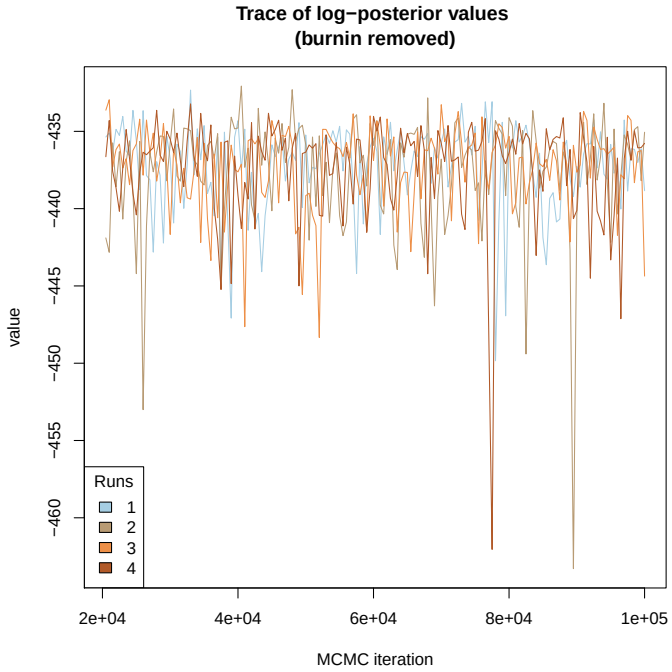
1.2 Assessing convergence and determining the burnin

A MCMC is said to converge when it reaches a stationary state, i.e. its distributional properties a constant over time (mean and variance don't depend on which iteration you consider). Convergence of the chains is best assessed by comparing parallel runs. This can be done using `plotChains`:

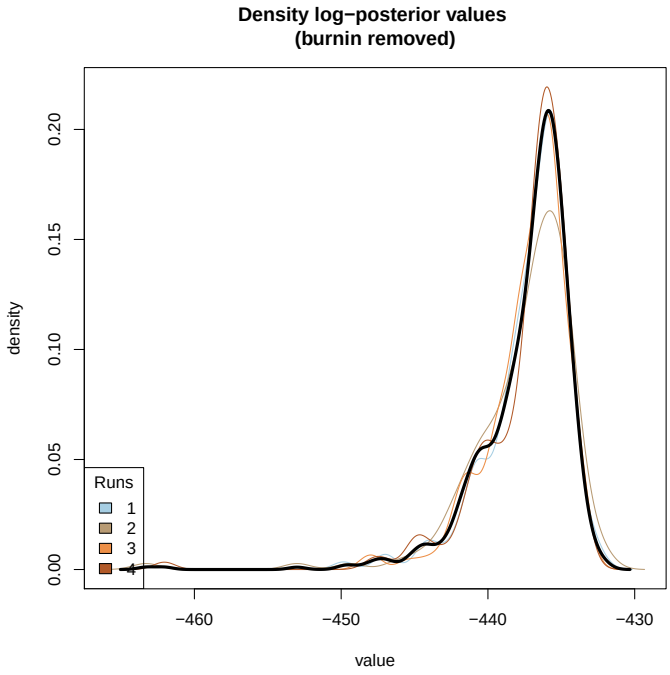
```
plotChains(res, main="Trace of log-posterior values")
```




```
plotChains(res, main="Trace of log-posterior values \n(burnin removed)",
           burnin=2e4)
```

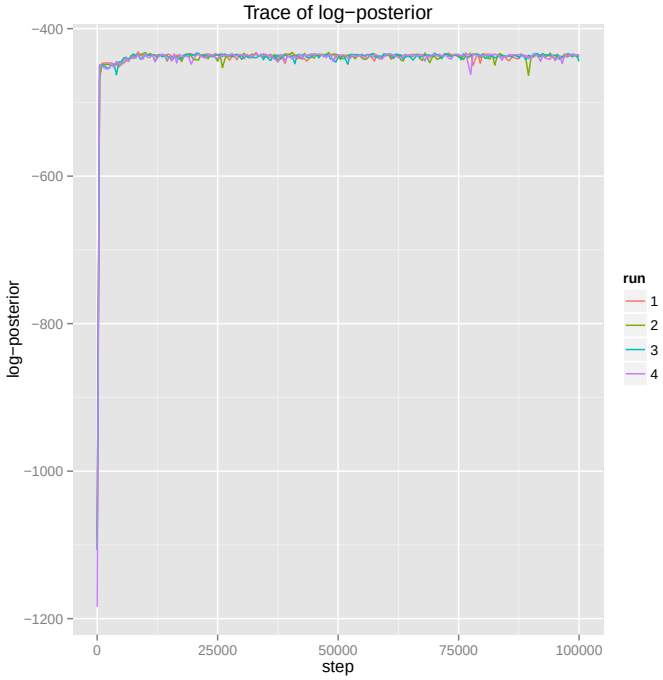



```
plotChains(res, main="Density log-posterior values \n(burnin removed)",
           burnin=2e4, type="dens")
```

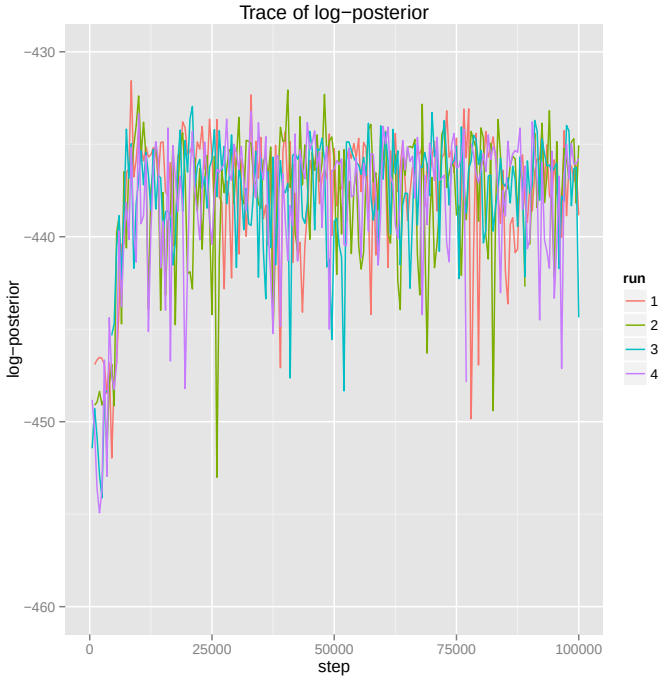



```
library(ggplot2)
library(reshape2)
x <- res$chains
x$run <- factor(x$run)
```

```
p <- ggplot(x, aes(x=step)) +
    geom_line(aes(y=post, colour=run)) +
    labs(title="Trace of log-posterior", y="log-posterior")
p
```

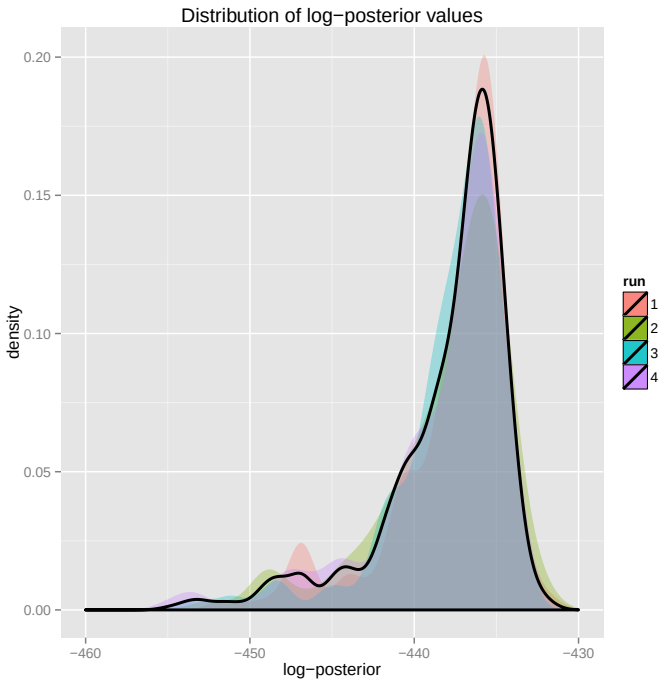



```
p + scale_y_continuous(limits=c(-460,-430))
```

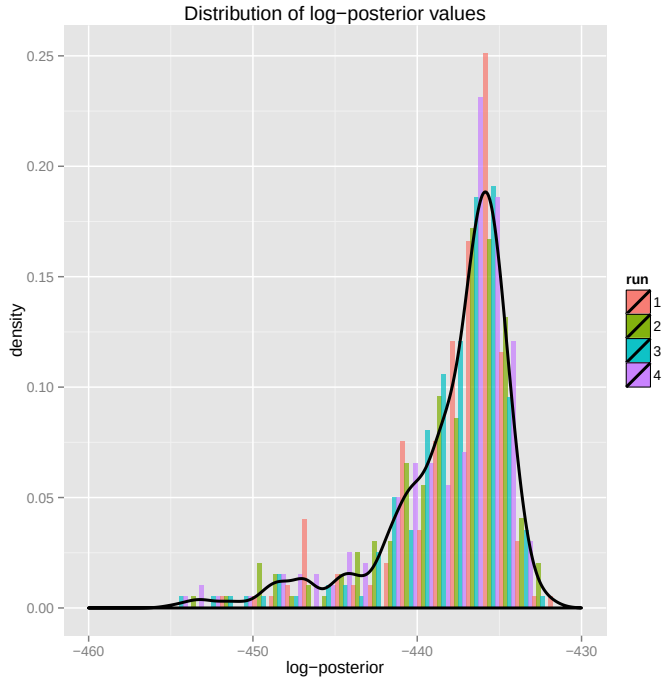



```
p + scale_y_continuous(limits=c(-460,-430)) + geom_smooth(aes(y=post))

## geom_smooth:  method="auto" and size of largest group is <1000, so using loess.
Use 'method = x' to change the smoothing method.
```


```
p <- ggplot(data=x) + labs(title="Distribution of log-posterior values", x="log-posterior") +
    scale_x_continuous(limits=c(-460,-430))
p + geom_density(aes(x=post, fill=run), alpha=.3, colour=NA) +
    geom_density(aes(x=post), size=1, colour="black", shape=2, alpha=.8)
```



```
p + geom_histogram(aes(x=post, fill=run, y=..density..), alpha=.7, colour=NA, position="dodge") +
    geom_density(aes(x=post), size=1, colour="black", shape=2, alpha=.8)

## stat_bin:  binwidth defaulted to range/30.  Use 'binwidth = x' to adjust this.
```

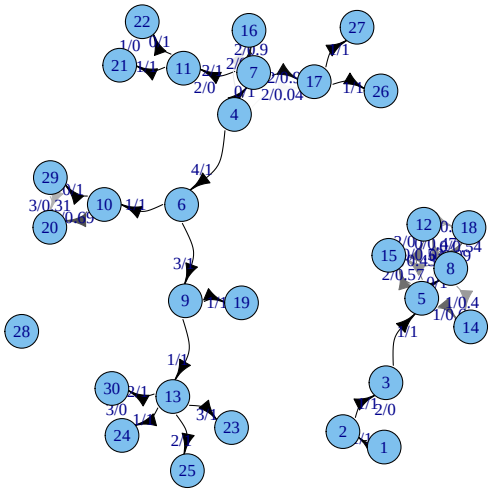


2 Interpreting the results

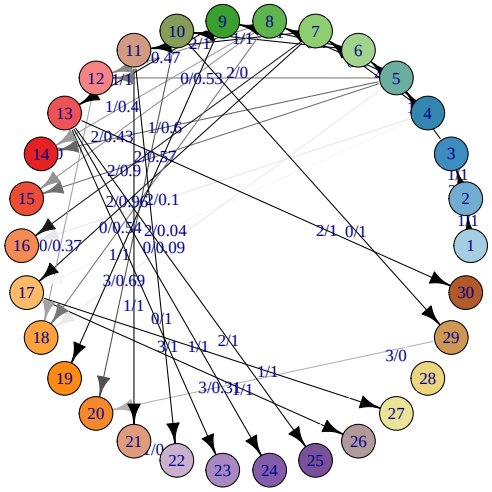
2.1 Visualizing reconstructed transmission trees

```
library(igraph)
library(adegenet)
```

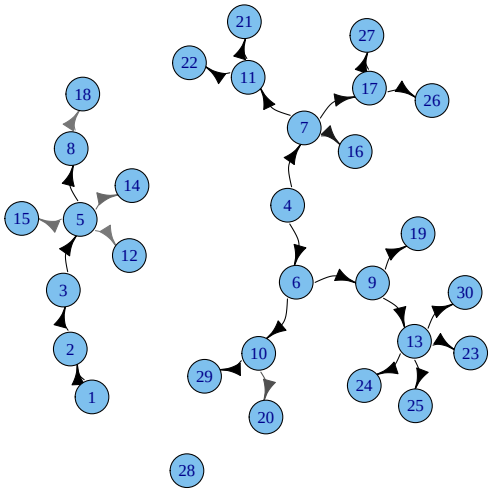
```
g <- transGraph(res, thres=0)
```



```
plot(g, layout=layout.circle, edge.curved=FALSE, vertex.color=funky(30))
```

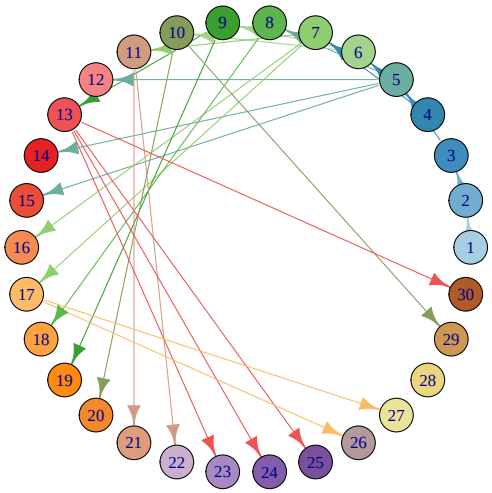


```
g <- transGraph(res, thres=0.5, annot="")
```

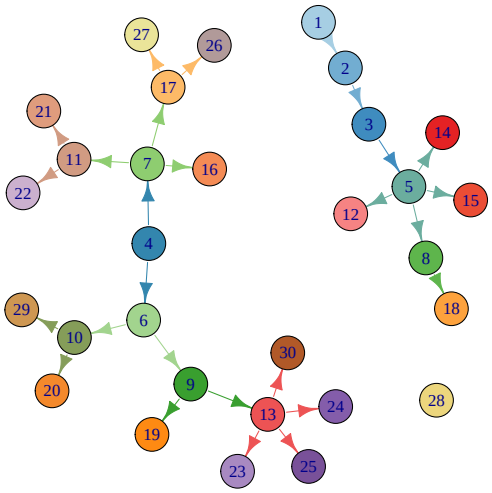



```
edge.colors <- funky(30)[as.numeric(get.edgelist(g)[,1])]
plot(g, layout=layout.circle, edge.curved=FALSE, vertex.color=funky(30),
     edge.color=edge.colors)
title("Ancestries with support >50% - circular graph")
```

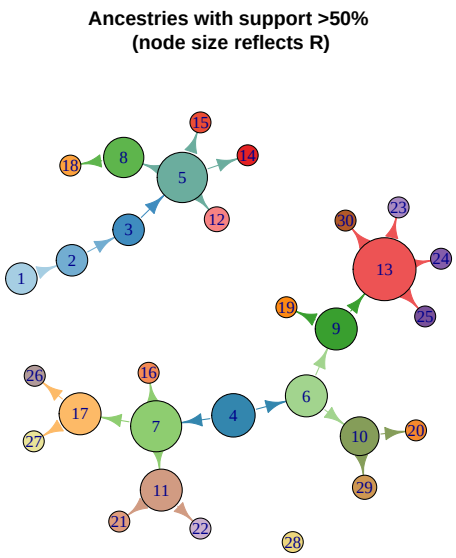
Ancestries with support >50% − circular graph



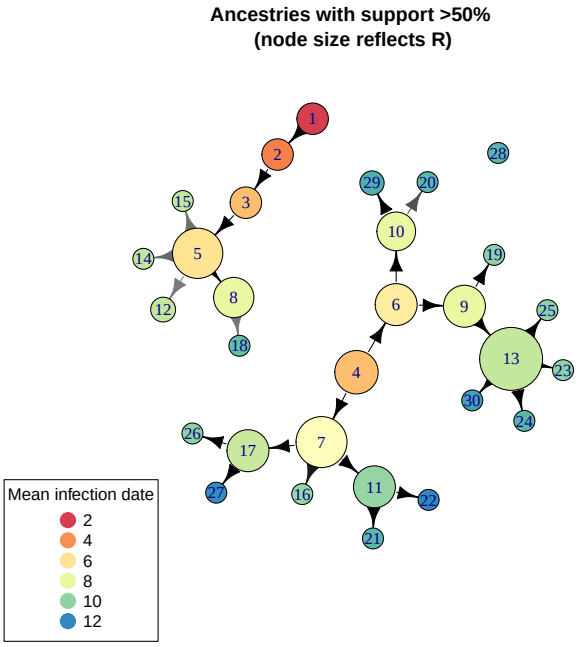
```
plot(g, layout=layout.auto, edge.curved=FALSE, vertex.color=funky(30),
     edge.color=edge.colors)
title("Ancestries with support >50% - other layout")
```



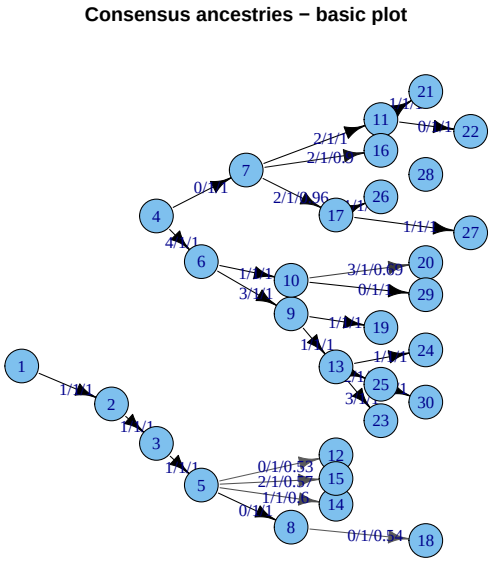

```
case.size <- 10+apply(get.R(res),2,mean)*5
plot(g, layout=layout.auto, edge.curved=FALSE, vertex.color=funky(30),
     edge.color=edge.colors, vertex.size=case.size)
title("Ancestries with support >50% \n(node size reflects R)")
```



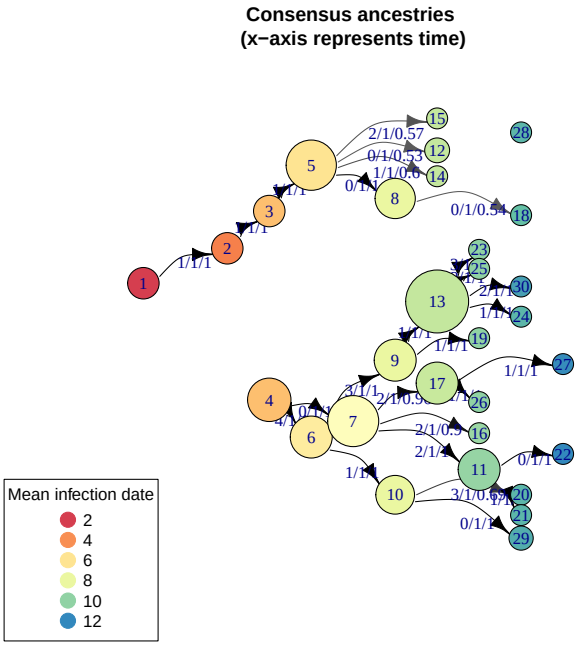

```
Tinf <- x[x$step>=2e4,grep("Tinf", names(x))]
case.color <- any2col(apply(Tinf,2,mean), col.pal=spectral)
plot(g, layout=layout.auto, edge.curved=FALSE, vertex.color=case.color$col,
     vertex.size=case.size)
title("Ancestries with support >50% \n(node size reflects R)")
legend("bottomleft", col=case.color$leg.col, leg=case.color$leg.txt, title="Mean infection date", pch
```



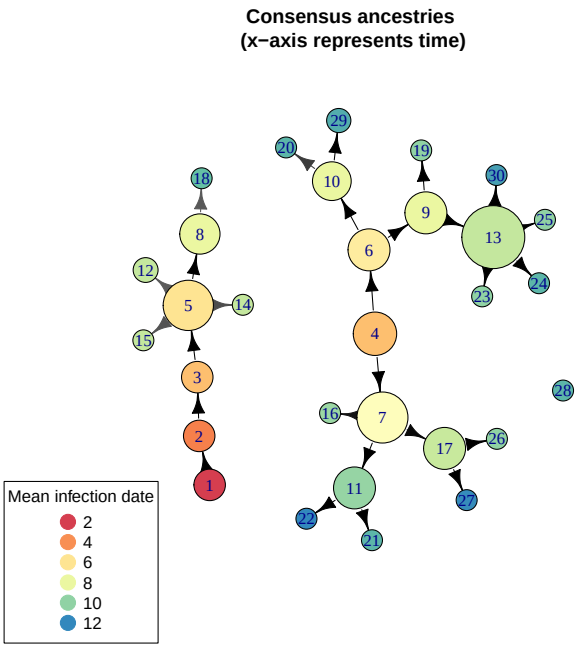
```
plot(get.tTree(res), main="Consensus ancestries - basic plot")
```




```
tre <- get.tTree(res)
plot(tre, edge.curved=TRUE, vertex.color=case.color$col,
     vertex.size=case.size)
title("Consensus ancestries \n(x-axis represents time)")
legend("bottomleft", col=case.color$leg.col, leg=case.color$leg.txt, title="Mean infection date", pch
```




```
g <- as.igraph(get.tTree(res))
plot(g, edge.curved=FALSE, vertex.color=case.color$col, layout=layout.auto, vertex.size=case.size, ed
title("Consensus ancestries \n(x-axis represents time)")
legend("bottomleft", col=case.color$leg.col, leg=case.color$leg.txt, title="Mean infection date", pch
```



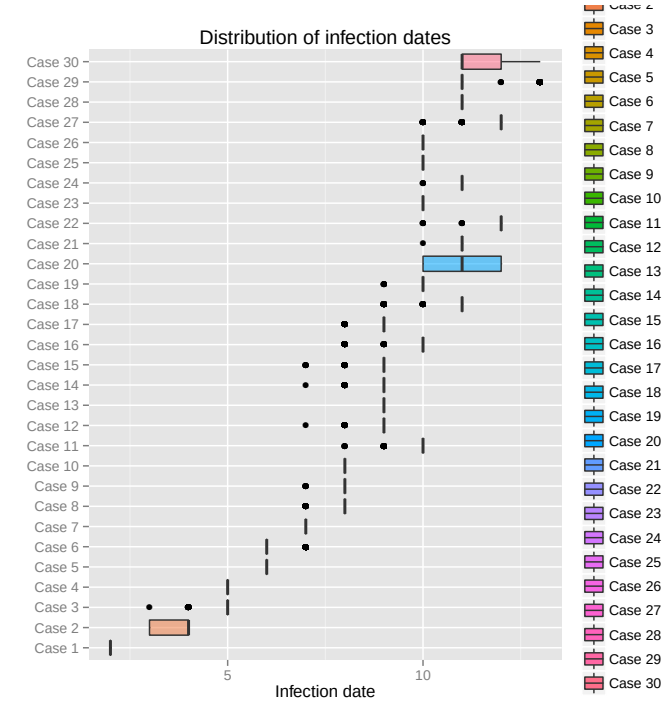
2.2 Plotting dates of infection

```
Tinf <- x[x$step>=2e4,c(1,ncol(x),grep("Tinf", names(x)))]
Tinf[1:5,1:6]

##     step run Tinf_1 Tinf_2 Tinf_3 Tinf_4
## 41 20000   1      2      4      5      5
## 42 20500   1      2      4      5      5
## 43 21000   1      2      4      5      5
## 44 21500   1      2      4      5      5
## 45 22000   1      2      3      5      5
```

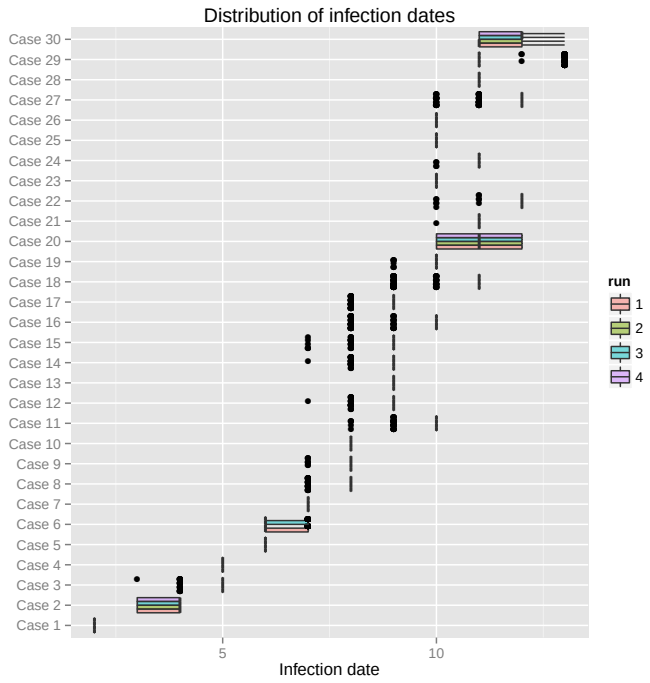
```
Tinf <- melt(Tinf, id=1:2)
names(Tinf)[3:4] <- c("case", "date")
Tinf$case <- sub("Tinf_","Case ", Tinf$case)
Tinf$case <- factor(Tinf$case, levels=paste("Case",1:30))
head(Tinf)

##    step run   case date
## 1 20000   1 Case 1    2
## 2 20500   1 Case 1    2
## 3 21000   1 Case 1    2
## 4 21500   1 Case 1    2
## 5 22000   1 Case 1    2
## 6 22500   1 Case 1    2

tail(Tinf)

##         step run    case date
## 19315  97500   4 Case 30   11
## 19316  98000   4 Case 30   12
## 19317  98500   4 Case 30   12
## 19318  99000   4 Case 30   12
## 19319  99500   4 Case 30   12
## 19320 100000   4 Case 30   11
```

```
ggplot(data=Tinf) + geom_line(aes(x=step,y=date,colour=case), alpha=.5) +
    labs(y="Infection date", title="Traces of infection dates")
```


```
ggplot(data=Tinf) + geom_boxplot(aes(x=case,y=date,fill=case, alpha=.5)) +
   coord_flip() + labs(y="Infection date", x="", title="Distribution of infection dates")
```



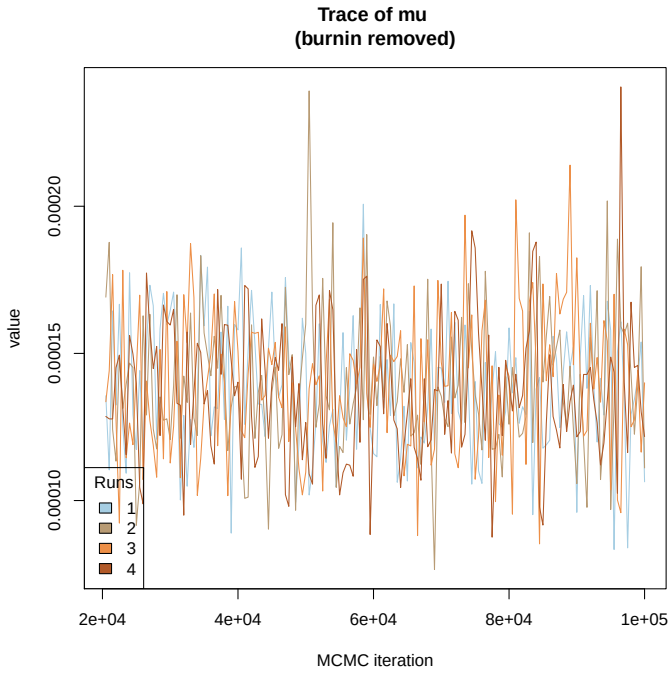

```
ggplot(data=Tinf) + geom_boxplot(aes(x=case,y=date,fill=run),alpha=.5) + coord_flip() +
    labs(y="Infection date", x="", title="Distribution of infection dates")
```



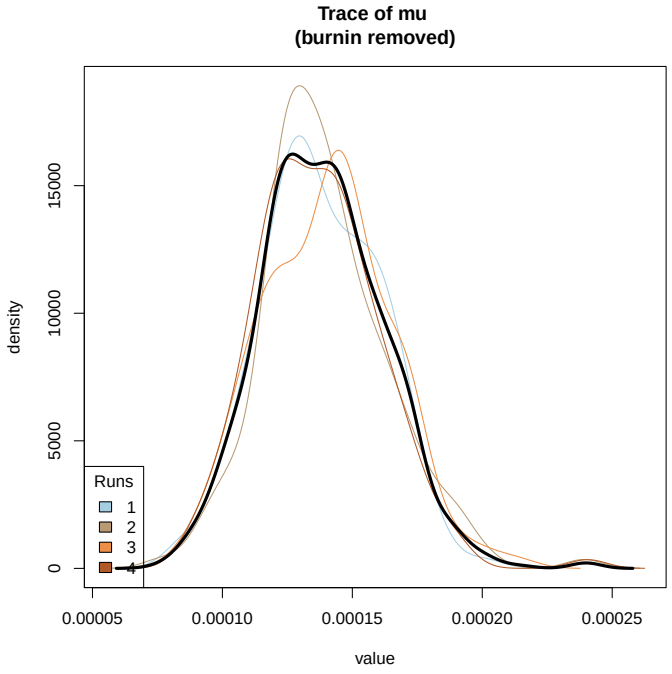
2.3 Accessing posterior distributions

Note that any element of the model in `res$chains` can be plotted using `plotChains`; for instance, the mutation rate:

```
plotChains(res, main="Trace of mu \n(burnin removed)",
           burnin=2e4, what="mu1")
```



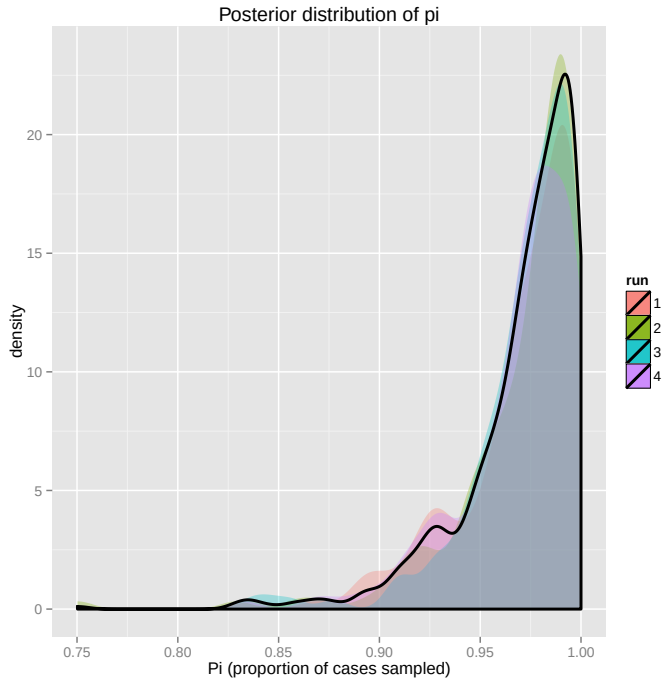

```
plotChains(res, main="Trace of mu \n(burnin removed)",
           burnin=2e4, what="mu1", type="dens")
```



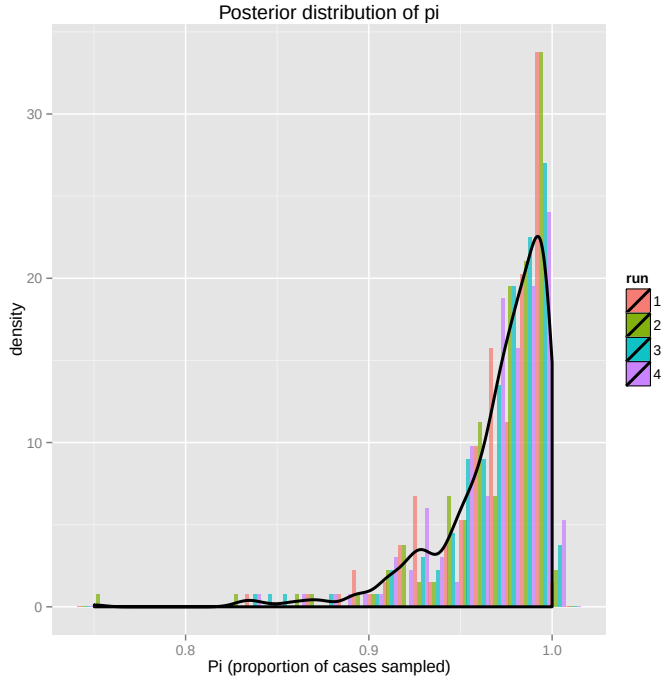
(note that in this case, the plotted information is the mutation rate *per generation of infection*, and not per unit of time. See section on mutation rates below for an estimation of the rates per unit of time.

```
library(ggplot2)
library(reshape2)
x <- res$chains[x$step>2e4,]
x$run <- factor(x$run)
```

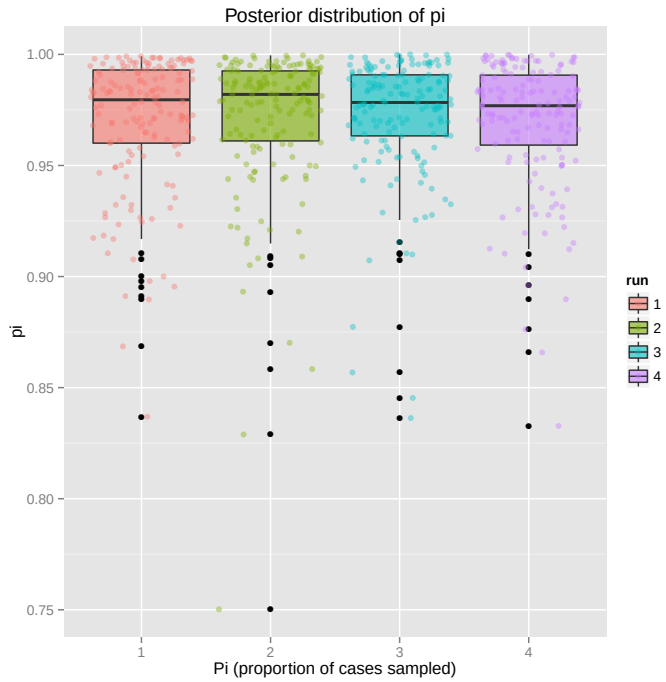
```
p <- ggplot(data=x) + labs(title="Posterior distribution of pi", x="Pi (proportion of cases sampled)"
p + geom_density(aes(x=pi, fill=run), alpha=.3, colour=NA) +
    geom_density(aes(x=pi), size=1, colour="black", shape=2, alpha=.8)
```



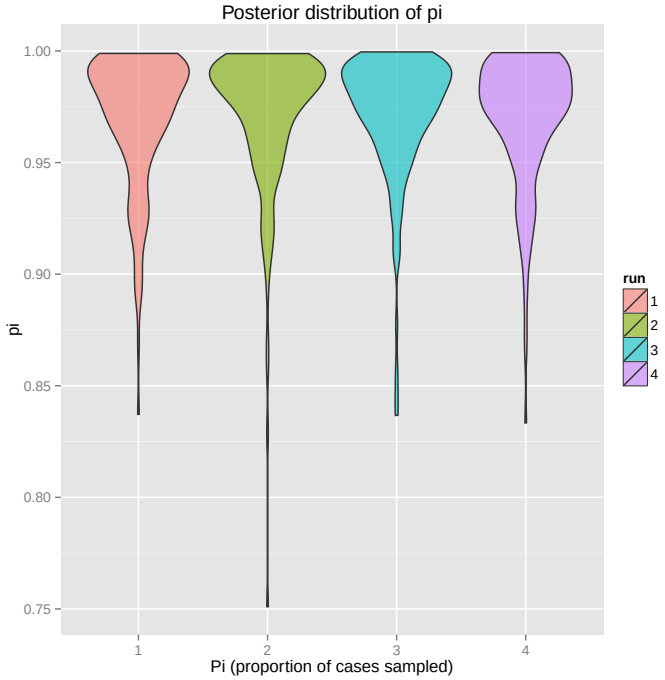

```
p + geom_histogram(aes(x=pi, fill=run, y=..density..), alpha=.7, colour=NA, position="dodge") +
    geom_density(aes(x=pi), size=1, colour="black", shape=2, alpha=.8)

## stat_bin:  binwidth defaulted to range/30.  Use 'binwidth = x' to adjust this.
```




```
p + geom_boxplot(aes(x=run, y=pi, fill=run),alpha=.6) + geom_jitter(aes(x=run, y=pi, col=run),alpha=.
```

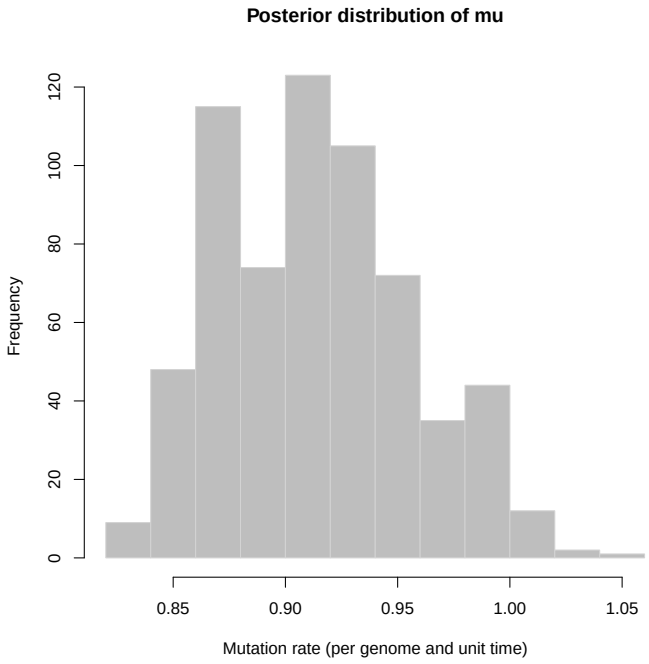



```
p + geom_violin(aes(x=run, y=pi, fill=run),alpha=.6)
```

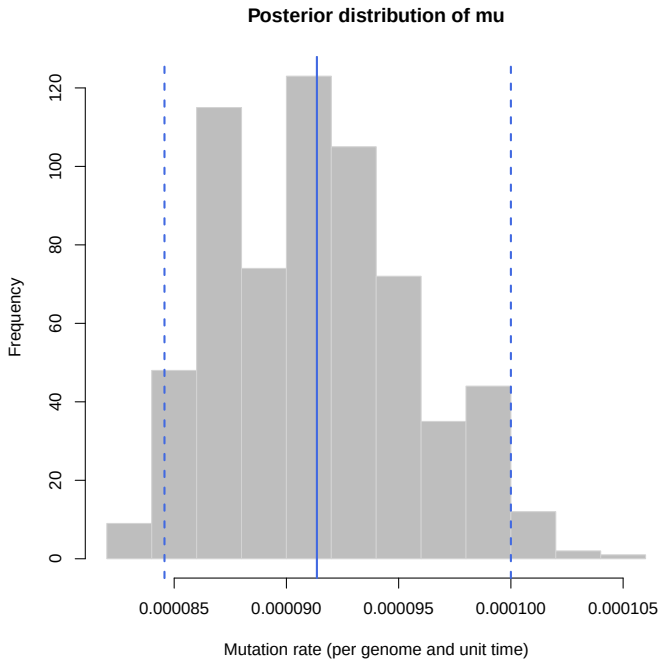


2.4 Mutation rates

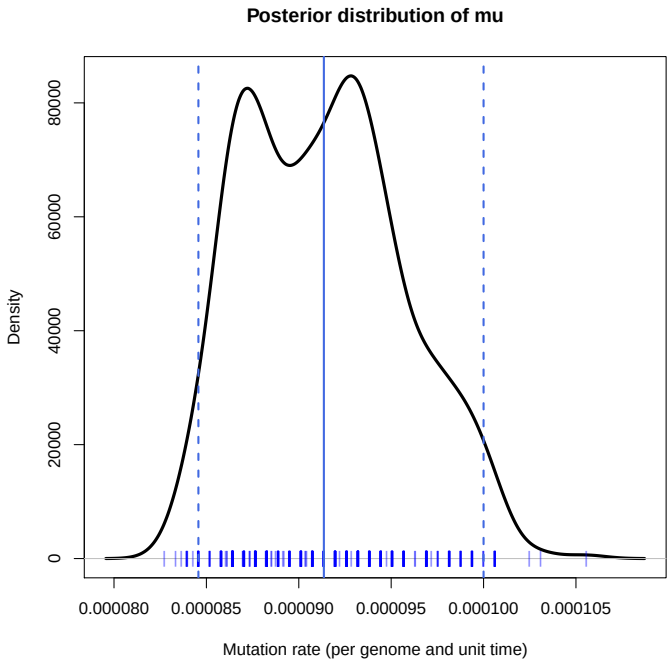
```
mu <- get.mu(res, burnin=2e4)
hist(mu, col="grey",border="lightgrey", xlab="Mutation rate (per genome and unit time)",
     main="Posterior distribution of mu")
```



```
mu <- get.mu(res, burnin=2e4, genome.size=ncol(dat$dna))
summary(mu)

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 8.27e-05 8.77e-05 9.14e-05 9.14e-05 9.44e-05 1.06e-04

hist(mu, col="grey",border="lightgrey", xlab="Mutation rate (per genome and unit time)",
     main="Posterior distribution of mu")
abline(v=quantile(mu, c(.025, .5, .975)), lty=c(2,1,2), lwd=2, col="royalblue")
```

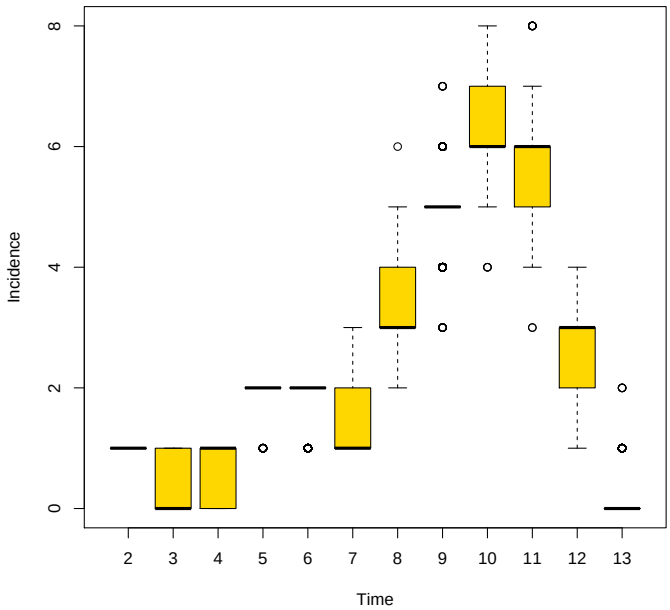



```
plot(density(mu), xlab="Mutation rate (per genome and unit time)",
     main="Posterior distribution of mu", lwd=3)
points(jitter(mu), rep(0, length(mu)), pch="|", col=transp("blue"))
abline(v=quantile(mu, c(.025, .5, .975)), lty=c(2,1,2), lwd=2, col="royalblue")
```

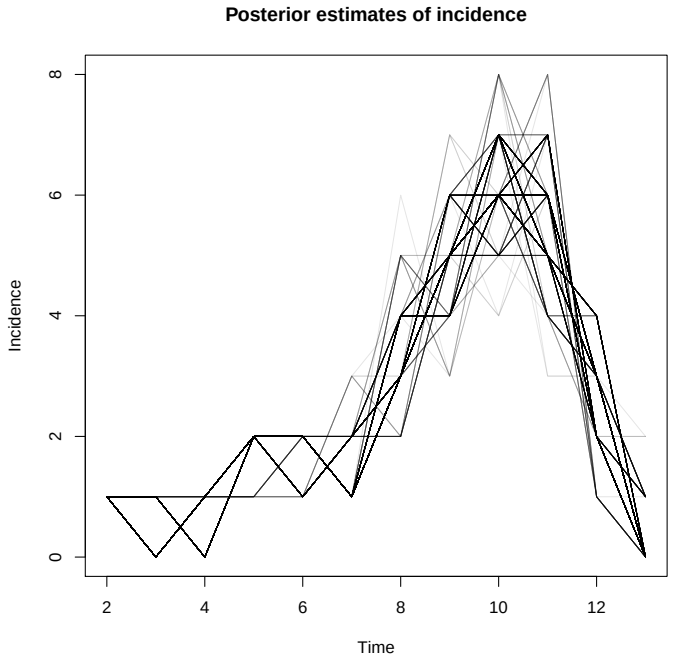


2.5 Incidence and reproduction numbers

```
incid <- get.incid(res)
```

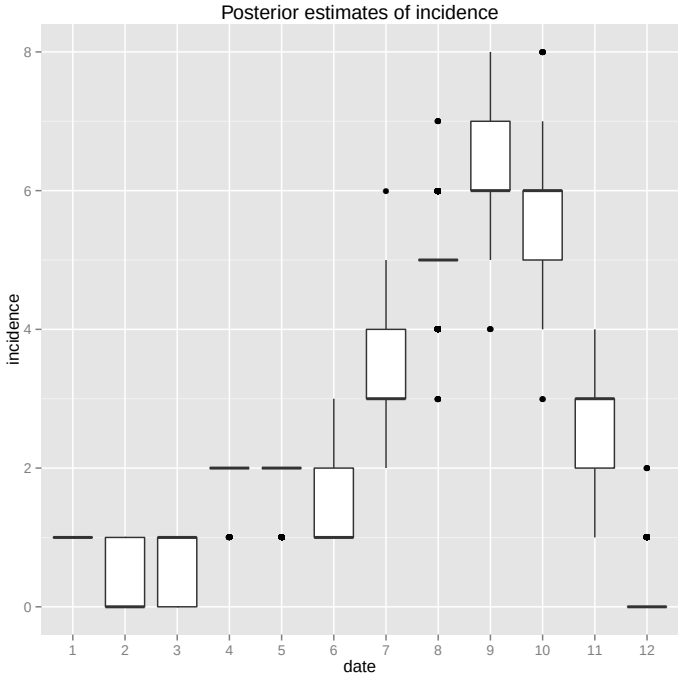


```
args(get.incid)

## function (x, burnin = 20000, plot = TRUE, type = c("boxplot",
##     "lines"), lines = FALSE, fill.col = "gold", lines.col = transp("grey"),
##     ...)
## NULL

incid <- get.incid(res, type="lines",lines.col=transp("black",.1))
title("Posterior estimates of incidence")
```

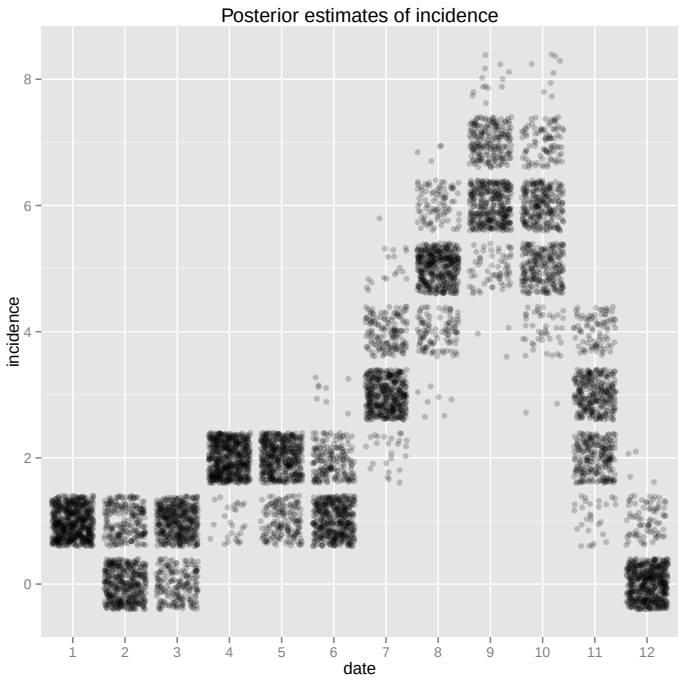



```
x <- data.frame(date=as.vector(row(incid)),
                step=as.vector(col(incid)),
                incidence=as.vector(incid))
head(x)

##   date step incidence
## 1    1    1         1
## 2    2    1         0
## 3    3    1         1
## 4    4    1         2
## 5    5    1         2
## 6    6    1         1

p <- ggplot(data=x, aes(x=date, y=incidence)) + labs(title="Posterior estimates of incidence")
```

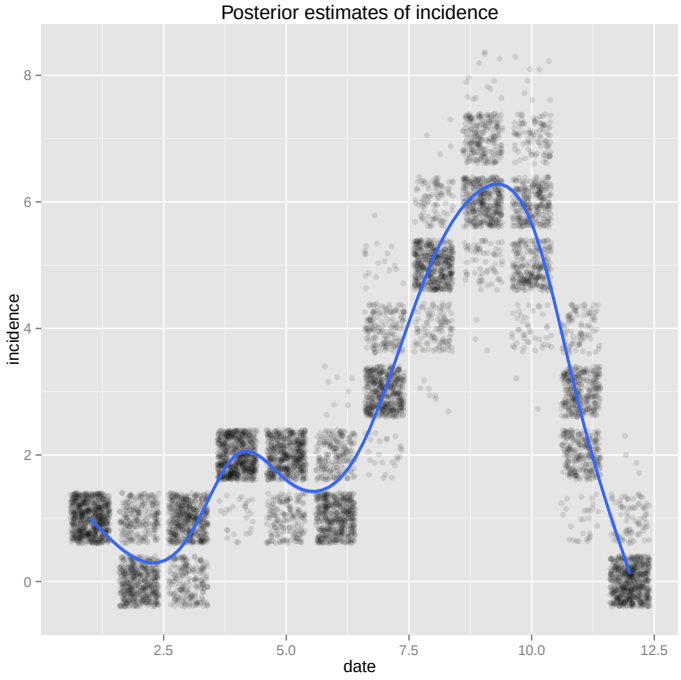
```
p + geom_boxplot(aes(x=factor(date)))
```



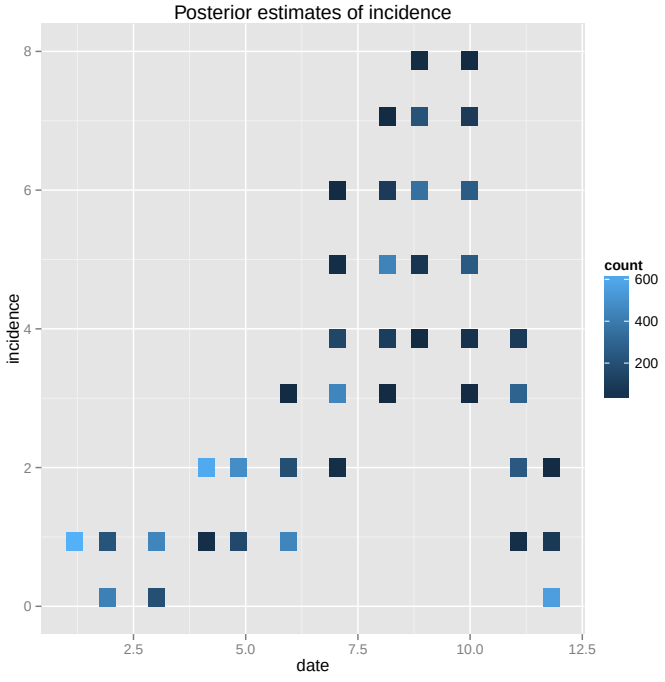

```
p + geom_jitter(aes(x=factor(date)), alpha=.2)
```



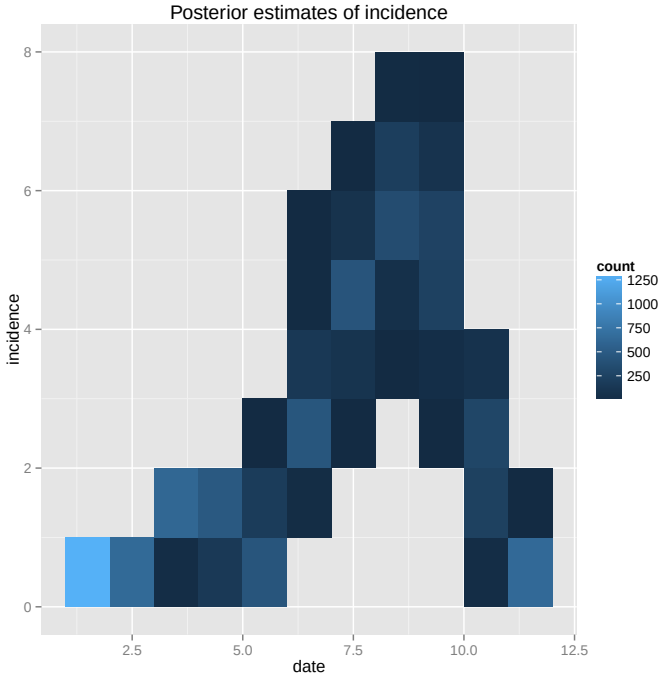

```
p + geom_jitter(alpha=.1) + geom_smooth(method=lm, formula=y~ns(x,10), size=1)
```



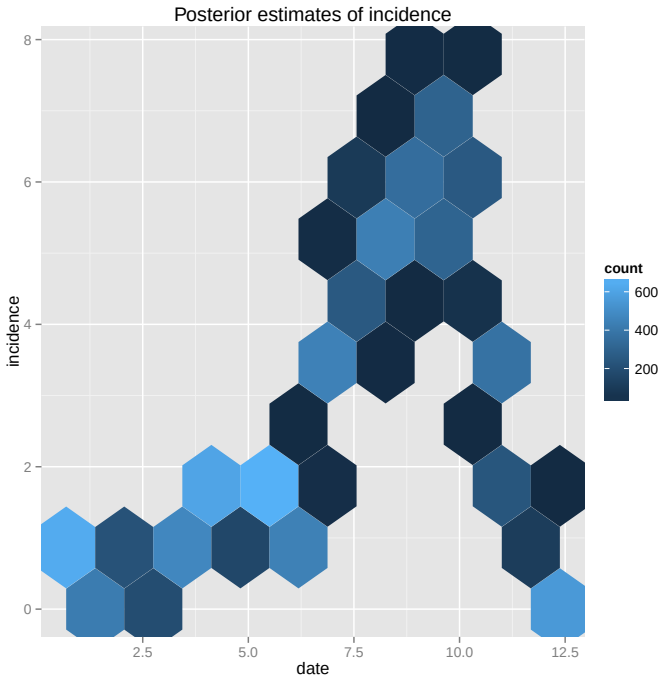

```
p + geom_bin2d()
```




```
p + geom_bin2d(binwidth=c(1,1))
```

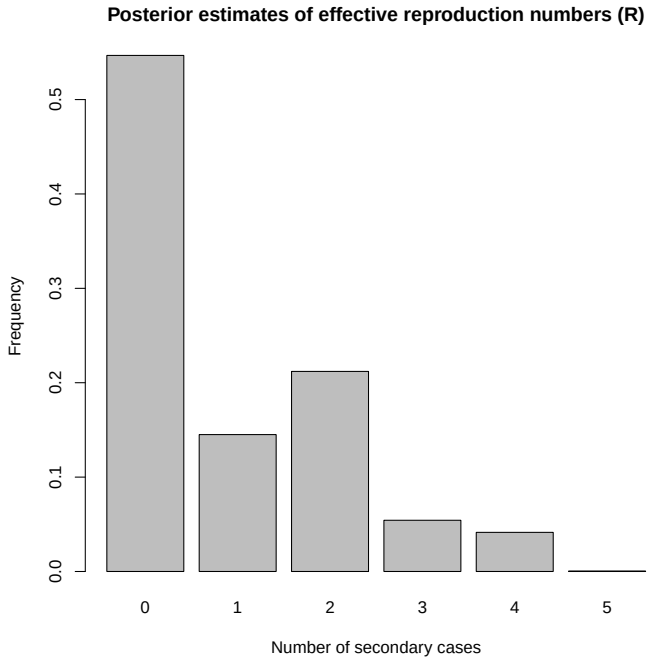



```
p + geom_hex(bins=8)
```



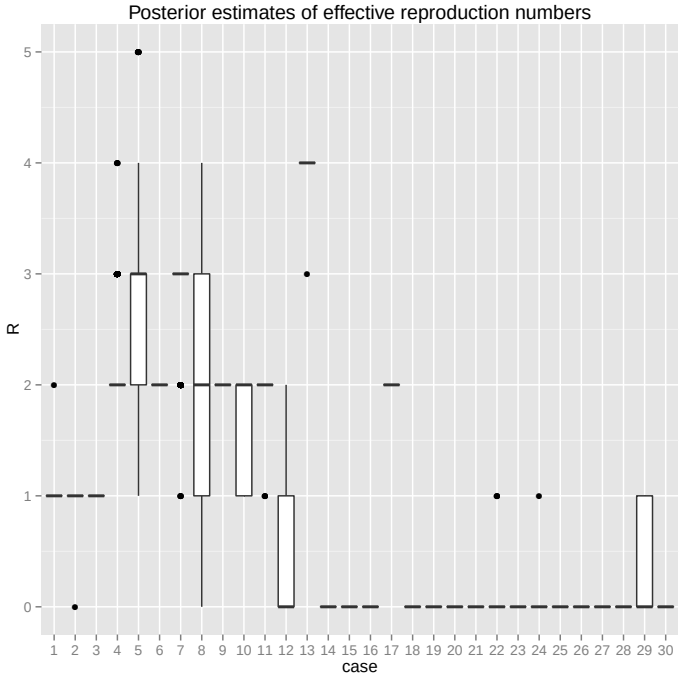

```
R <- get.R(res)
dim(R)

## [1] 640  30
```

```
barplot(table(R)/length(R), xlab="Number of secondary cases", main="Posterior estimates of effective
```




```
x <- data.frame(case=factor(as.vector(col(R)), levels=as.character(1:30)),R=as.vector(R))
head(x)

##   case R
## 1    1 1
## 2    1 1
## 3    1 1
## 4    1 1
## 5    1 1
## 6    1 1

tail(x)

##       case R
## 19195   30 0
## 19196   30 0
## 19197   30 0
## 19198   30 0
## 19199   30 0
## 19200   30 0
```

```
p <- ggplot(data=x, aes(x=case, y=R)) + labs(title="Posterior estimates of effective reproduction num
p + geom_boxplot()
```




```
p + geom_boxplot(aes(colour=case))
```