Analyze_transmission

Preamble

Title: *Script B for Objective 2: effects of seasonal network differences on pathogen transmission*
Author: Marie Gilbertson
Date: "08/30/2020"

What this code does:
1. Analyzes differences between wet and dry season epidemics

```
#### Clear Environment ####
remove(list=ls())

#### Load libraries ####
library(ggplot2)
library(dplyr)
library(stringr) # for "str_split"
library(viridis)
library(ggpubr)

#### Set seed ####
set.seed(3268)
```

Load data

Start by loading parameter sets and preparing to load data of interest.

```
# load parameter sets
param.sets <- get(load("../Data/simulation_paramsets.Rdata"))

# set which parameter sets to analyze
param.ids <- c(1:420)

# empty dataframe to store results
all.results <- NULL
summ.stats <- data.frame(matrix(nrow = length(param.ids), ncol = 7))
colnames(summ.stats) <- c("param.set", "m.dur.time", "m.total.i",
"m.final.i",
                          "m.max.i", "m.max.it_s", "prop.fail")
```

Next, loop through parameter sets of interest, loading those simulation results.

```
# loop through and load all results
for(i in 1:length(param.ids)){
  param.set.num <- param.ids[i]

  full.name <- paste("../Output/Simulation_Results/full set results_paramset
", param.set.num, ".Rdata", sep = "")
  set_results <- get(load(full.name))

  all.results <- rbind(all.results, set_results)

  #### store summary stats ####
  summ.stats$param.set[i] <- set_results$param.set[1]

  # proportion failed epidemics; if only 1 individual is ever infected,
consider it a "failed" epidemic
  bad.draws <- sum(set_results$num.failed)
  only.1 <- nrow(set_results[set_results$total.i<=(1/33),])
  # number of bad draws + number only infecting 1 / simulations + number of
bad draws
  summ.stats$prop.fail[i] <- (bad.draws + only.1)/(nrow(set_results) +
bad.draws)

  ## now, of only the "successful" epidemics:
  set_results_succ <- set_results[set_results$total.i>(1/33),]
  # mean duration
  summ.stats$m.dur.time[i] <- mean(set_results_succ$dur.time)
  # mean total ever infected
  summ.stats$m.total.i[i] <- mean(set_results_succ$total.i)
  # mean final infected
  summ.stats$m.final.i[i] <- mean(set_results_succ$final.i)
  # mean max infected
  summ.stats$m.max.i[i] <- mean(set_results_succ$max.i)
  # mean start point of max infected
  summ.stats$m.max.it_s[i] <- mean(set_results_succ$max.it_s)
}
```

Update formatting, subset, and save for ease in plotting later.

```
# assign appropriate model type and gamma labeling for plotting
param.sets$model.type <- NA
param.sets$gamma <- NA
for(i in 1:nrow(param.sets)){
  temp.params <- param.sets[i,]
  model.type_gamma <- unlist(str_split(temp.params$model.type_gamma, "_"))
  param.sets$model.type[i] <- model.type_gamma[1]  # type of disease process;
options are "SI", "SIR", or "SIS"
```

```
  if(model.type_gamma[1]=="SI"){
    param.sets$gamma[i] <- 0
  }else{
    param.sets$gamma[i] <- as.numeric(model.type_gamma[2])
  }
}

param.sets$param.set <- paste("set_", param.sets$set.num, sep = "")
summ.stats <- left_join(summ.stats, param.sets, by = "param.set")
all.results <- left_join(all.results, param.sets, by = "param.set")

# split data and save as subsets of interest
weight.data <- subset(summ.stats, summ.stats$weights==T)
binary.data <- subset(summ.stats, summ.stats$weights==F)

w.si <- subset(weight.data, weight.data$model.type=="SI")
b.si <- subset(binary.data, binary.data$model.type=="SI")

w.sis <- subset(weight.data, weight.data$model.type=="SIS")
b.sis <- subset(binary.data, binary.data$model.type=="SIS")

w.sir <- subset(weight.data, weight.data$model.type=="SIR")
b.sir <- subset(binary.data, binary.data$model.type=="SIR")

### save these datasets for making manuscript figures ###
# save(w.si, file =
"../Output/Manuscript_Figures/data_for_plotting/w.si.Rdata")
# save(b.si, file =
"../Output/Manuscript_Figures/data_for_plotting/b.si.Rdata")
# save(w.sis, file =
"../Output/Manuscript_Figures/data_for_plotting/w.sis.Rdata")
# save(b.sis, file =
"../Output/Manuscript_Figures/data_for_plotting/b.sis.Rdata")
# save(w.sir, file =
"../Output/Manuscript_Figures/data_for_plotting/w.sir.Rdata")
# save(b.sir, file =
"../Output/Manuscript_Figures/data_for_plotting/b.sir.Rdata")
```

Automated generation of results heatmaps

Loop through all model types, generating results heatmaps for epidemic outcomes of interest.

```
model.types <- c("w.si", "b.si" , "w.sir", "b.sir", "w.sis", "b.sis")
```

```r
for(i in 1:length(model.types)){

  temp.model <- model.types[i]
  print(temp.model)


  # read in data
  name <- paste("../Output/Manuscript_Figures/data_for_plotting/",
temp.model, ".Rdata", sep = "")
  temp.data <- get(load(name))

  temp.data.d <- subset(temp.data, temp.data$season=="Dry")
  temp.data.w <- subset(temp.data, temp.data$season=="Wet")
  temp.data_diff <- temp.data.d
  temp.data_diff[,c("m.dur.time", "m.total.i","prop.fail")] <-
temp.data.d[,c("m.dur.time", "m.total.i","prop.fail")] -
temp.data.w[,c("m.dur.time", "m.total.i","prop.fail")]



  #### MEAN DURATION ####
  if(temp.model %in% c("w.sir", "b.sir", "w.sis", "b.sis")){

    g1_st <- ggplot(temp.data, aes(x = c.rate, y = prob)) +
geom_tile(aes(fill = m.dur.time), colour = "white") +
      scale_fill_viridis(discrete=F, option = "B", direction = 1, limits =
c(1, 26)) +
      ylab("Probability of transmission") +
      xlab("Edge weight scaling")


    gg1_st <- g1_st + facet_grid(gamma ~ season) +
      theme(text = element_text(size=15)) +
      theme(axis.text.x = element_text(size = 10),
            legend.position = "bottom") +
      labs(fill = "Mean duration (weeks)")
    gg1_st

    # differences
    g2_st <- ggplot(temp.data_diff, aes(x = c.rate, y = prob)) +
geom_tile(aes(fill = m.dur.time), colour = "white") +
      scale_fill_gradient2(low = "blue", mid = "white", high = "red",
midpoint = 0, breaks = c(-2, 0, 2, 4), limits = c(-2.6, 4.4)) +
      ylab("Probability of transmission") +
      xlab("Edge weight scaling")


    gg2_st <- g2_st + facet_grid(gamma~.) +
      theme(text = element_text(size=15)) +
```

```
      theme(axis.text.x = element_text(size = 10),
            legend.position = "bottom") +
      labs(fill = "Dry mean difference (weeks)")
    gg2_st


    g <- ggarrange(gg1_st, gg2_st, labels = c("A", "B"), nrow = 1, widths =
c(1.5, 1))
    g

    dur.name <- paste("../Output/Manuscript_Figures/",
temp.model,"_duration_full figure.jpeg", sep = "")
    ggsave(g, file = dur.name, width = 11, height = 11, units = "in", dpi =
300)
  }

  #### MEAN TOTAL INFECTED ####
  g1_st <- ggplot(temp.data, aes(x = c.rate, y = prob)) + geom_tile(aes(fill
= m.total.i), colour = "white") +
    scale_fill_viridis(discrete=F, option = "B", direction = 1, limits =
c(0.06, 1)) +
    ylab("Probability of transmission") +
    xlab("Edge weight scaling")

  gg1_st <- g1_st + facet_grid(gamma ~ season) +
    theme(text = element_text(size=15)) +
    theme(axis.text.x = element_text(size = 10),
          legend.position = "bottom", legend.key.width = unit(1, "cm")) +
    labs(fill = "Mean total proportion infected")
  gg1_st


  g2_st <- ggplot(temp.data_diff, aes(x = c.rate, y = prob)) +
geom_tile(aes(fill = m.total.i), colour = "white") +
    scale_fill_gradient2(low = "blue", mid = "white", high = "red", midpoint
= 0, breaks = c(-0.08, 0, 0.08, 0.16), limits = c(-0.08, 0.2)) +
    ylab("Probability of transmission") +
    xlab("Edge weight scaling")

  gg2_st <- g2_st + facet_grid(gamma~.) +
    theme(text = element_text(size=15)) +
    theme(axis.text.x = element_text(size = 10),
          legend.position = "bottom", legend.key.width = unit(1, "cm")) +
    labs(fill = "Dry mean difference")
  gg2_st

  g <- ggarrange(gg1_st, gg2_st, labels = c("A", "B"), nrow = 1, widths =
c(1.5, 1))
  g
```

```
  if(temp.model %in% c("w.si", "b.si")){
    total.name <- paste("../Output/Manuscript_Figures/", temp.model,"_total
infected_full figure.jpeg", sep = "")
    ggsave(g, file = total.name, width = 11, height = 5, units = "in", dpi =
300)
  }else{
    total.name <- paste("../Output/Manuscript_Figures/", temp.model,"_total
infected_full figure.jpeg", sep = "")
    ggsave(g, file = total.name, width = 11, height = 11, units = "in", dpi =
300)
  }


  #### MEAN PROPORTION FAILED ####

  g1_st <- ggplot(temp.data, aes(x = c.rate, y = prob)) + geom_tile(aes(fill
= prop.fail), colour = "white") +
    scale_fill_viridis(discrete=F, option = "B", direction = -1, limits =
c(0, 1)) +
    ylab("Probability of transmission") +
    xlab("Edge weight scaling")

  gg1_st <- g1_st + facet_grid(gamma ~ season) +
    theme(text = element_text(size=15)) +
    theme(axis.text.x = element_text(size = 10),
          legend.position = "bottom", legend.key.width = unit(1, "cm")) +
    labs(fill = "Proportion failed")
  gg1_st




  g2_st <- ggplot(temp.data_diff, aes(x = c.rate, y = prob)) +
geom_tile(aes(fill = prop.fail), colour = "white") +
    scale_fill_gradient2(low = "red", mid = "white", high = "blue", midpoint
= 0, breaks = c(-0.2, -0.1, 0, 0.1), limits = c(-0.22, 0.12)) +
    ylab("Probability of transmission") +
    xlab("Edge weight scaling")

  gg2_st <- g2_st + facet_grid(gamma~.) +
    theme(text = element_text(size=15)) +
    theme(axis.text.x = element_text(size = 10),
          legend.position = "bottom") +
    labs(fill = "Dry difference")
  gg2_st
```

```
  g <- ggarrange(gg1_st, gg2_st, labels = c("A", "B"), nrow = 1, widths =
c(1.5, 1))
  g

  if(temp.model %in% c("w.si", "b.si")){
    fail.name <- paste("../Output/Manuscript_Figures/", temp.model,"_prop
failed_full figure.jpeg", sep = "")
    ggsave(g, file = fail.name, width = 11, height = 5, units = "in", dpi =
300)
  }else{
    fail.name <- paste("../Output/Manuscript_Figures/", temp.model,"_prop
failed_full figure.jpeg", sep = "")
    ggsave(g, file = fail.name, width = 11, height = 11, units = "in", dpi =
300)
  }

}
```

```
## [1] "w.si"
```

```
## [1] "b.si"
```

```
## [1] "w.sir"
```

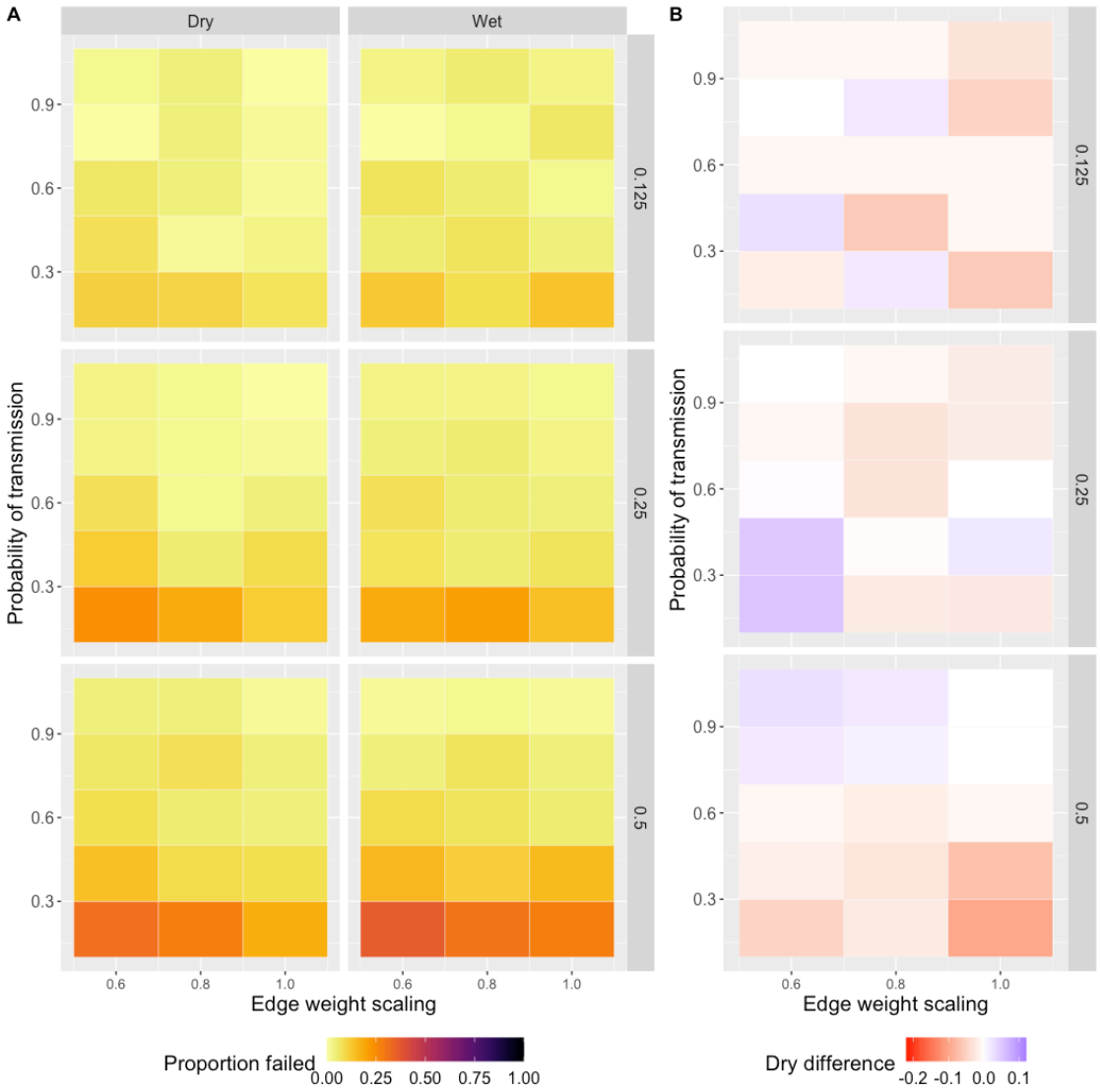
```
## [1] "b.sir"
```

```
## [1] "w.sis"
```

```
## [1] "b.sis"
```

For illustration purposes, view example output.

```
print(g)
```



For reproducibility, view session info.

```
sessionInfo()
```

```
## R version 3.6.3 (2020-02-29)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS  10.16
##
## Matrix products: default
## BLAS:
/Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib
## LAPACK:
/Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib
##
## locale:
```

```
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] ggpubr_0.4.0      viridis_0.5.1     viridisLite_0.3.0 stringr_1.4.0
## [5] dplyr_1.0.4       ggplot2_3.3.3
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.1.0  xfun_0.24         purrr_0.3.4       haven_2.2.0
##  [5] carData_3.0-4     colorspace_2.0-0  vctrs_0.3.6       generics_0.1.0
##  [9] htmltools_0.5.1.1 yaml_2.2.1        rlang_0.4.10      pillar_1.4.7
## [13] foreign_0.8-76    glue_1.4.2        withr_2.4.1       DBI_1.1.1
## [17] readxl_1.3.1      lifecycle_1.0.0   munsell_0.5.0     ggsignif_0.6.0
## [21] gtable_0.3.0      cellranger_1.1.0  zip_2.1.0         evaluate_0.14
## [25] labeling_0.4.2    knitr_1.28        rio_0.5.16        forcats_0.5.0
## [29] curl_4.3          broom_0.7.2       Rcpp_1.0.6        scales_1.1.1
## [33] backports_1.1.6   abind_1.4-5       farver_2.0.3      gridExtra_2.3
## [37] hms_1.0.0         digest_0.6.27     stringi_1.5.3     openxlsx_4.1.5
## [41] rstatix_0.6.0     cowplot_1.1.0     grid_3.6.3        tools_3.6.3
## [45] magrittr_2.0.1    tibble_3.0.6      crayon_1.4.1      tidyr_1.0.2
## [49] car_3.0-9         pkgconfig_2.0.3   ellipsis_0.3.1
data.table_1.12.8
## [53] rmarkdown_2.9     R6_2.5.0          compiler_3.6.3
```