Plots produced for How to downsize a complex society (Environmental Research Letters, 2020)

Andreas Angourakis

15 January 2020

This file and all other referenced in the code can be found at the repository: https://github.com/Andros-Spica/ERL_Angourakis-et-al-2020_Rproject

Figure 3. Daily precipitation and seasonal totals (mm) for on the location of the five major Indus urban centres.

Using:

> Daniel Wallach, David Makowski, James W. Jones, François Brun (2019), 'Working with dynamic crop models: Methods, tools, and examples for agriculture and enviromnent', Academic Press.

Model description in p. 24-28, R code example in p. 116-122. Another, possibly newer version of the R code can be found at: https://github.com/cran/ZeBook/blob/master/R/watbal.model.r

Load functions from 'Soil Water Balance model' library file:

```
source("source/watbal.model.R")
```

Technical note: The ZeBook package, watbal.model.R has a bug in lines 21-25 and 58. The problem is that the code fails to extract values from the param object generated by `watbal.define.param` because it does not account for both dimensions of this object: (WHC, MUF, DC, z, CN) x (nominal, binf, bsup). I solve this problem by adding "typeOfParameterValue" as an argument in watbal.model that is passed also to watbal.update.

Load function to estimate reference evapotranspiration:

```
source("source/estimateETr.R")
```

Load CurlyBraces function for plotting:

```
source("source/CurlyBraces.R")
```

< begin parentesis >

To reproduce Figure 4.5 in Wallach et al. 2006, load weather daily data and select site and year :

```
weather <- watbal.weather(working.year = 2008, working.site = 1)
```

Or following code on the book:

```
weather <- ZeBook::weather_FranceWest
weather <- weather[(weather$idsite == 1 & weather$WEYR == 2008),]
```

And skip to the steps related to the Soil Water Balance model (bellow).

< end parentesis >

To generate Figure 3, we use the data downloaded at NASA´s POWER access viewer (power.larc.nasa.gov/data-access-viewer/) selecting the user community 'Agroclimatology' and pin pointing the five different locations between 01/01/1984 and 31/12/2018. The exact locations are:

- Mohenjo-daro (Latitude: 27.3242, Longitude: 68.1367)
- Dholavira (Latitude: 23.8776 Longitude: 70.2184)
- Ganweriwala (Latitude: XX, Longitude: XX)
- Harappa (Latitude: 30.6333 Longitude: 72.8667)
- Rakhigarhi (Latitude: 29.1687, Longitude: 76.0687)

We selected the ICASA Format's parameters:

- Precipitation (PRECTOT)
- Wind speed at 2m (WS2M)
- Relative Humidity at 2m (RH2M)
- Dew/frost point at 2m (T2MDEW)
- Maximum temperature at 2m (T2M_MAX)
- Minimum temperature at 2m (T2M_MIN)
- All sky insolation incident on a horizontal surface (ALLSKY_SFC_SW_DWN)
- Temperature at 2m (T2M)

The following code reads all files inside the "Fig3_input" folder and creates a single data frame that contains the data of all five locations:

```
inputFiles <- paste0("Fig3_input/", list.files(path = "Fig3_input"))

weather <- data.frame()

for (i in 1:length(inputFiles))
{
  tempData <- watbal.weather.file(inputFiles[i], year = 1984:2008)

  # convert any missing data in all sky insolation (I) from -99.0 to Na
  tempData$I[tempData$I == -99] <- NA

  weather <- rbind(weather, tempData)
}

rm(tempData)

# use Latitude to distinguish sites and create a new variable using site names:
weather$Site <- rep(NA, nrow(weather))
for (i in 1:nrow(weather))
{
  if (floor(weather$LAT[i]) == 27) { weather$Site[i] <- "Mohenjo-daro" }
```

```
  if (floor(weather$LAT[i]) == 23) { weather$Site[i] <- "Dholavira" }
  if (floor(weather$LAT[i]) == 28) { weather$Site[i] <- "Ganweriwala" }
  if (floor(weather$LAT[i]) == 30) { weather$Site[i] <- "Harappa" }
  if (floor(weather$LAT[i]) == 29) { weather$Site[i] <- "Rakhigarhi" }
}
weather$Site <- factor(weather$Site,
                       levels = c("Mohenjo-daro", "Ganweriwala", "Harappa",
                                  "Dholavira", "Rakhigarhi"))
```

Sum up yearly and seasonal totals by splitting the year into summer (May-Oct or from day) and winter (Nov-Apr):

```
precipitationSums <- list(Site = c(), year = c(), total = c(), summer = c(), winter = c())

for (site in levels(weather$Site))
{
  for (year in levels(factor(weather$year)))
  {
    precipitationSums$Site <- c(precipitationSums$Site, site)
    precipitationSums$year <- c(precipitationSums$year, year)

    precipitationSums$total <- c(precipitationSums$total,
                                 round(sum(weather$RAIN[weather$Site == site &
                                                        weather$year == year])))
    precipitationSums$winter <- c(precipitationSums$winter,
                                  round(sum(weather$RAIN[weather$Site == site &
                                                         weather$year == year &
                                                         (weather$day <= 121 | weather$day > 305)]
                                      )))
    precipitationSums$summer <- c(precipitationSums$summer,
                                  round(sum(weather$RAIN[weather$Site == site &
                                                         weather$year == year &
                                                         (weather$day > 121 & weather$day <= 305)]
                                      )))
  }
}

precipitationSums <- data.frame(precipitationSums)
```

Calculate the average annual precipitation in every site to reference in the text:

```
for (site in levels(precipitationSums$Site))
{
  cat(paste0(site, "\n"))
  print(summary(precipitationSums$total[precipitationSums$Site == site], na.rm = TRUE))
  cat("----------------------------------------\n")
}
```

```
## Dholavira
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    55.0   188.0   258.0   303.3   420.0   671.0
## ----------------------------------------
## Ganweriwala
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      30      85     107     117     149     219
```

```
## ----------------------------------------
## Harappa
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     123     164     232     240     299     459
## ----------------------------------------
## Mohenjo-daro
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    8.00   33.00   72.00   78.96   98.00  286.00
## ----------------------------------------
## Rakhigarhi
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     248     392     495     499     615     740
## ----------------------------------------
```

Estimate reference evapotranspiration using FAO recomendations (Penman-Monteith method):

```
weather$ETr <- estimateETr(weather$I,
                           weather$T2M,
                           weather$Tmax,
                           weather$Tmin,
                           weather$T2MDEW,
                           weather$WS2M)
```

Use Soil Water Balance model to calculate soil water content proportion or WATp (mm mm-1 day-1) and ARID coefficient.

The first step is to set all soil parameters, here assumed to be constants throughout the sites:

```
parameters <- watbal.define.param()
# This function creates the following configuration (from Wallach et al. 2006):
# WHC = 0.15
# MUF = 0.096
# DC = 0.55
# z = 40
# CN = 65

# input variables describing soil (from Wallach et al. 2006)
soil.WP = 0.06
soil.FC = soil.WP + parameters["nominal", "WHC"]
```

Run the model inputting parameters and dataset:

```
aWatbal <- cbind(Site = weather$Site,
                 watbal.model(parameters,
                        weather,
                        WP = soil.WP,
                        FC = soil.FC,
                        typeOfParValue = "nominal"))
```

`watbal.model`iterates for every day in the dataset calculating soil water content, absolute and proportion (WAT, WATp), and the respective ARID coefficient.

Plot precipitation (RAIN), evapotranspiration (ETr), soil water content proportion (WATp), and the ARID coefficient for the five sites during the year 1990.

```
selectedYear = 1990
lastDayOfWinter = 121
lastDayOfSummer = 305
```

```r
plotName = "Fig3.png"

png(plotName, width = 1000, height = 600)

layout(matrix(1:25, nrow = 5, ncol = 5),
       heights = c(0.2, 1.1, 1, 1, 1.3),
       widths = c(1.25, 1, 1, 1, 1))

yLabs <- c("Precipitation (mm)", "ETr (mm)", "WATp (%)", "ARID index")
leftMargin <- 4.1

# x coordinate in barplots has different scale, so must be adjusted with this offset
curlyBracesOffset = 1.2

for (site in levels(factor(weather$Site)))
{
  if (site != levels(factor(weather$Site))[1])
  {
    yLabs <- rep("", 4)
    leftMargin <- 2.1
  }
  tempData <- aWatbal[aWatbal$year == selectedYear & aWatbal$Site == site,]

  par(mar = c(0, leftMargin, 0, 0), cex = 1.2)

  plot(c(0, 1), c(0, 1), ann = F, bty = 'n', type = 'n', xaxt = 'n', yaxt = 'n')
  text(x = 0.5, y = 0.7, font = 4, cex = 1.2,
      labels = site)
  text(x = 0.5, y = 0.1, cex = 0.8,
       labels = paste0(precipitationSums$total[precipitationSums$Site == site &
                                                 precipitationSums$year == selectedYear],
                       " mm (annual total)")

       )

  par(mar = c(1.1, leftMargin, 2.1, 0.2))

  barplot(tempData$RAIN,
          ylab = yLabs[1])
  CurlyBraces(x0 = (lastDayOfWinter + 1) * curlyBracesOffset,
              x1 = lastDayOfSummer * curlyBracesOffset,
              y0 = 1.1 * max(tempData$RAIN), y1 = 1.1 * max(tempData$RAIN),
              position = 3,
              bracesSize = 0.15 * max(tempData$RAIN),
              label = paste0(
                precipitationSums$summer[precipitationSums$Site == site &
                                                 precipitationSums$year == selectedYear],
                " mm (summer)"),
              labelSize = 1, labelDist = 1.1)
  CurlyBraces(x0 = 1 * curlyBracesOffset,
              x1 = lastDayOfWinter * curlyBracesOffset,
              y0 = -0.1 * max(tempData$RAIN), y1 = -0.1 * max(tempData$RAIN),
              position = 1,
```

```
              bracesSize = 0.15 * max(tempData$RAIN),
              label = paste0(
                precipitationSums$winter[precipitationSums$Site == site &
                                           precipitationSums$year == selectedYear],
                " mm (winter)"),
              labelSize = 1, labelDist = 1, labelAdj = 0)
  CurlyBraces(x0 = (lastDayOfSummer + 1) * curlyBracesOffset,
              x1 = 365 * curlyBracesOffset,
              # x1 coordinate must be higher than 365 to cover the rest of the axis
              y0 = -0.1 * max(tempData$RAIN), y1 = -0.1 * max(tempData$RAIN),
              position = 1,
              bracesSize = 0.15 * max(tempData$RAIN))
  par(mar = c(0.1, leftMargin, 1.1, 0.2))

  barplot(tempData$ETr,
          ylab = yLabs[2])

  plot(1:nrow(tempData), tempData$WATp * 100,
     xlab = "", xaxt = 'n',
     ylab = yLabs[3],
     type = "l", lwd = 2, ylim = c(0, soil.FC * 110))
  abline(h = soil.FC * 100, lty = 2)
  abline(h = soil.WP * 100, lty = 2)

  par(mar = c(3.5, leftMargin, 1.1, 0.2))

  plot(1:nrow(tempData), tempData$ARID,
       xlab = "day\n", ylab = yLabs[4],
       type = "l", lwd = 2, ylim = c(0, 1))
}
rm(tempData)

dev.off()
```

```
## pdf
##   2
```

```
knitr::include_graphics(plotName)
```

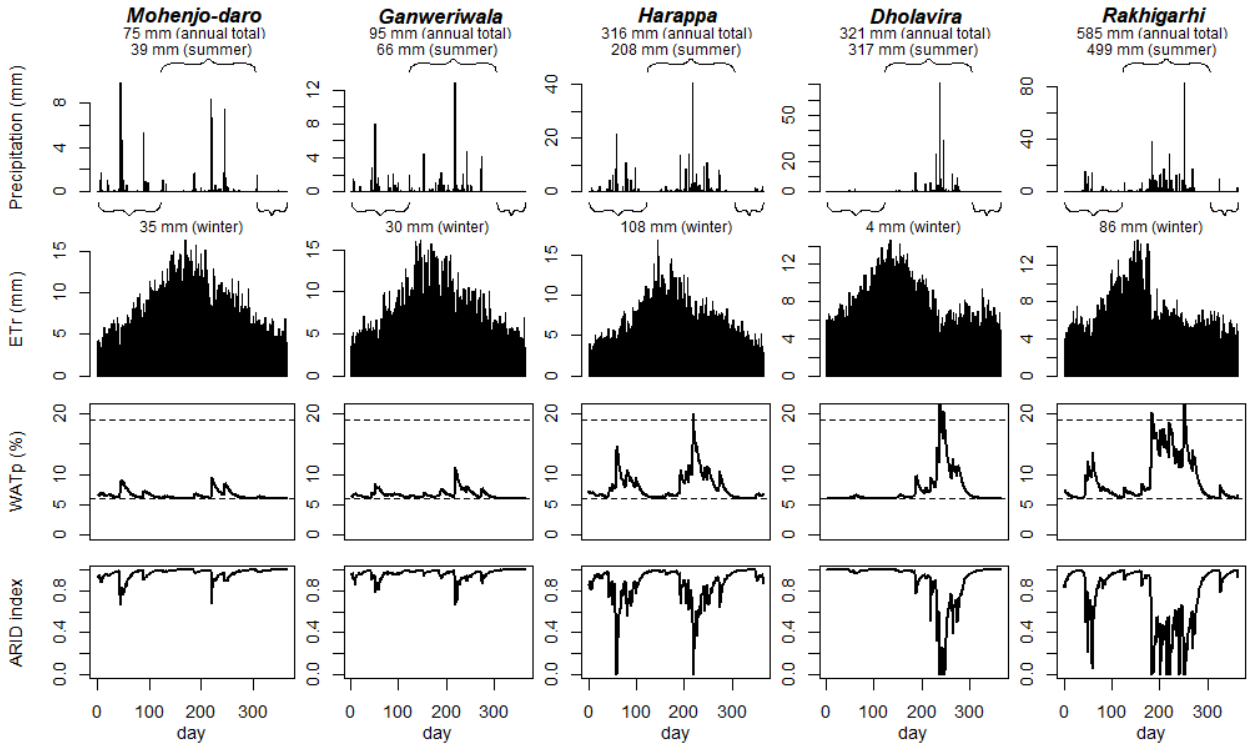

Figure 6. Example of simulated weather variables

Load output generated with NetLogo implementation of the SIMPLE crop model, which integrates the Weather and Soil Water Balance models:

```
simExample <- read.csv("Fig6_input/SIMPLE-crop-model experiment-table.csv", skip = 6)

wheatHarvestDay = simExample$"item.0.sugHarvestingDay"[1]
riceHarvestDay = simExample$"item.1.sugHarvestingDay"[1]


simExample <- simExample[, c(
  "X.step.", "T", "RAIN", "solarRadiation", "ET_0",
  "mean..WATp..of.patches", "mean..ARID..of.patches",
  "mean..biomass..of.patches.with..position.crop.typesOfCrops...0.",
  "mean..biomass..of.patches.with..position.crop.typesOfCrops...1.",
  "mean..yield..of.patches.with..position.crop.typesOfCrops...0.",
  "mean..yield..of.patches.with..position.crop.typesOfCrops...1."
)]

names(simExample) <- c(
  "day", "T", "RAIN", "solarRadiation", "ETr",
  "WATp", "ARID",
  "wheat_biomass", "rice_biomass", "wheat_yield", "rice_yield"
  )

# create vector with harves yields per crop and year

years <- 1:floor(nrow(simExample) / 365) - 1
```

```
cropYields <- data.frame(
  wheatHarvestDays = (years * 365 + wheatHarvestDay),
  riceHarvestDays = (years * 365 + riceHarvestDay),
  wheat = unique(simExample$wheat_yield),
  rice = unique(simExample$rice_yield)[-1])
cropYields$wheat[1] <- NA # ignore first zero yield in wheat
```

Load function for marking the end of each year:

```
markEndYears <- function(lengthOfData, offset = 1){
  for (i in 1:lengthOfData)
  {
    if (i %% (365 * offset) == 0)
    {
      abline(v = i, lty = 3)
    }
  }
}
```

Plot all selected variables:

```
plotName = "Fig6.png"

png(plotName, width = 1000, height = 1200)

layout(matrix(1:8, nrow = 8, ncol = 1),
       heights = c(1, 1, 1, 1, 1, 1, 1.3, 0.2))

yLabs <- c(expression(paste("Solar Radiation (", MJ/m^-2, ")")),
           "Temperature (ºC)",
           "Precipitation (mm)",
           "ETr (mm)", "WATp (%)", "ARID index")
leftMargin <- 5.1


# 1. Solar radiation
par(mar = c(0.1, leftMargin, 1.1, 0.2), cex.lab = 1.5)

plot(1:nrow(simExample), simExample$solarRadiation, type = "l",
     xlab = "", xaxt = 'n',
     ylab = yLabs[1])
markEndYears(nrow(simExample))

# 2. Temperature
plot(1:nrow(simExample), simExample$"T", type = "l",
     xlab = "", xaxt = 'n',
     ylab = yLabs[2])
markEndYears(nrow(simExample))

# 3. Precipitation
par(mar = c(1.1, leftMargin, 2.1, 0.2))

barplot(simExample$RAIN,
        ylab = yLabs[3])
markEndYears(nrow(simExample), offset = 1.2) ; abline(v = 2190 * 1.2, lty = 3)
```

```r
# not sure why, but barplot() x coordinates do not behave as in plot()

# 4. Reference evapotranspiration
par(mar = c(0.1, leftMargin, 1.1, 0.2))

barplot(simExample$ETr,
        ylab = yLabs[4])
markEndYears(nrow(simExample), offset = 1.2) ; abline(v = 2190 * 1.2, lty = 3)
# not sure why, but barplot() x coordinates do not behave as in plot()

# 5. Soil water content (%)
plot(1:nrow(simExample), simExample$WATp * 100,
     xlab = "", xaxt = 'n',
     ylab = yLabs[5],
     type = "l", lwd = 2, ylim = c(0, soil.FC * 110))
abline(h = soil.FC * 100, lty = 2)
abline(h = soil.WP * 100, lty = 2)
markEndYears(nrow(simExample))

# 6. ARID index
plot(1:nrow(simExample), simExample$ARID,
     ylab = yLabs[6],
     type = "l", lwd = 2, ylim = c(0, 1))
markEndYears(nrow(simExample))

# 7. Crop biomass and yield
par(mar = c(3.5, leftMargin, 1.1, 0.2))

plot(1:nrow(simExample), simExample$wheat_biomass, type = "l",
     xlab = "day\n", col = "darkred", lwd = 2,
     ylab = expression(paste("Biomass (", g/m^-2, ")")))
lines(1:nrow(simExample), simExample$rice_biomass, col = "darkblue", lwd = 2)
lines(cropYields$wheatHarvestDays, cropYields$wheat, col = "darkred", lwd = 2, lty = 3)
lines(cropYields$riceHarvestDays, cropYields$rice, col = "darkblue", lwd = 2, lty = 3)
markEndYears(nrow(simExample))

# 8. Legend (crops)
par(mar = c(0,0,0,0), cex = 1.2)

plot(c(0, 1), c(0, 1), ann = F, bty = 'n', type = 'n', xaxt = 'n', yaxt = 'n')

legend(x = 0.25, y = 0.99,
       legend = c('Triticum aestivum (Yecora Rojo)', 'Oryza sativa (IR72)'),
       col = c('darkred', 'darkblue'), lty = c(1, 1), lwd = 2,
       horiz = T, bty = "n")

dev.off()
```

```
## pdf
##   2
```

```r
knitr::include_graphics(plotName)
```

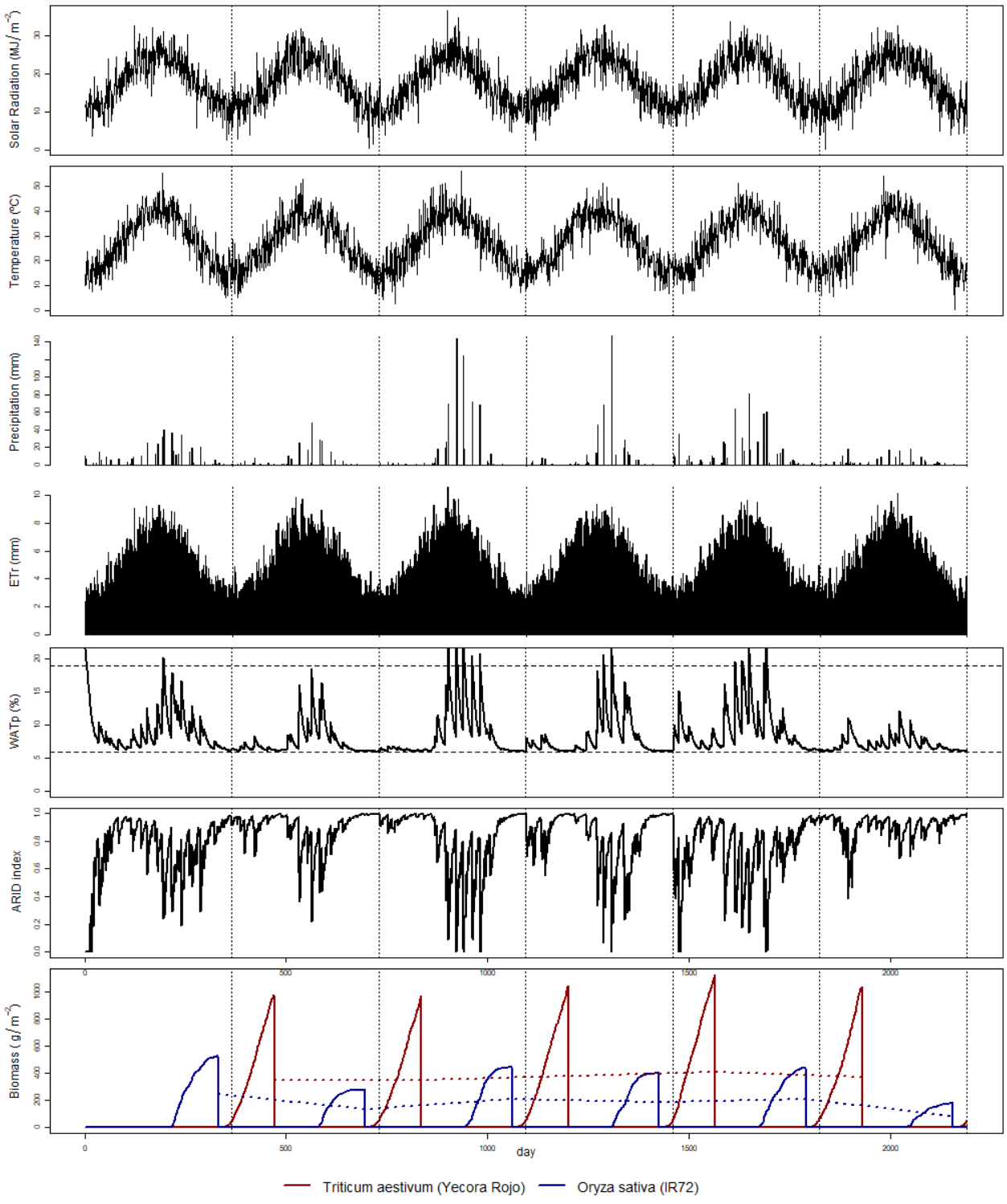

Figure 9. Example of response of the three life table models to parameter variation

Load functions that create/import Life Table models for fertility and nuptiality (Peristera-Kostaki model), and mortality (Coale-Demeny model):

```r
### fertility
# The following correspond to the first parametric model
# mentioned in page 147 of:
# # Peristeva and Kostaki, 2009
# "Modeling fertility in modern populations"
# Available from: https://dx.doi.org/10.4054/DemRes.2007.16.6

### nuptiality
# The following correspond to the first parametric model
# for fitting the age-specific distributions of marriages, mentioned in page 133 of:
# # Peristeva and Kostaki, 2015
# "A parametric model for estimating nuptiality patterns in modern populations"
# Available from: https://www.researchgate.net/publication/285457704_A_parametric_model_for_estimating_

# the equation is the same for both fertility and nuptiality.

generatePeristeraKostaki <- function(par.c1 = 0.8, par.mu = 25, par.sigma = c(10, 10) )
{
  curve <- c()
  for (i in 1:100)
  {
    sigma = par.sigma[1]
    if (i > par.mu)
    { sigma = par.sigma[2] }

    curve <- c(curve, par.c1 * exp (-1 * (((i - par.mu) / sigma) ^ 2)) )
  }

  return(curve)
}

### mortality
# from Coale-Demeny Model Life Tables generated with 'demoR' package
# demoR package version 0.6.0 (2018-09-13)
# by James Holland Jones and collaborators
# Their source:
# Coale, A., P. Demeny, and B. Vaughn. 1983.
# Regional model life tables and stable populations.
# 2nd ed. New York: Academic Press.

interpolatePerYear <- function(raw, ages = c(0.5, 1.5, 4, seq(8.5, 93.5, 5))) {

  perYear <- data.frame(matrix(numeric(0), nrow = 151, ncol = ncol(raw)))
  names(perYear) <- 1:ncol(perYear)

  for (i in 1:ncol(raw)) {
    perYear[, i] <- approx(ages, raw[, i],
                           xout = 1:151, yleft = 0, yright = 1)$y
  }
  row.names(perYear) <- 0:150

  return(perYear)
}
```

```
generateCoaleDemenyLifeTable <- function(region = "north", sex = "F", level = 8){

  curve <- 0

  if (region == "north")
  {
    curve <- interpolatePerYear(t(demogR::cdmltn(sex = sex)$nqx))[,level]
  }
  if (region == "west")
  {
    curve <- interpolatePerYear(t(demogR::cdmltw(sex = sex)$nqx))[,level]
  }
  if (region == "east")
  {
    curve <- interpolatePerYear(t(demogR::cdmlte(sex = sex)$nqx))[,level]
  }
  if (region == "south")
  {
    curve <- interpolatePerYear(t(demogR::cdmlts(sex = sex)$nqx))[,level]
  }

  return(curve)
}
```

Plot the curves generated by small explorations of the parameter space of each model:

```
grScale = 2

plotName = "Fig9.png"

png(plotName, width = grScale * 600, height = grScale * 600)

layout(matrix(1:2, nrow = 2, ncol = 1))

par(cex = grScale * 1.2, family = "serif",
    mar = c(3.5, 4.1, 1.1, 0.2))

# 1. Peristera-Kostaki equation (Fertility and Nuptiality models)

plot(c(1, 100), c(0, 1), type = "n",
     main = "Peristera-Kostaki equation",
     xlab = "AGE\n",
     ylab = "p(x)"
)

# five variations of parameter settings
parValues <- rbind(
  c(0.8, 25, 10, 10),
  c(0.95, 20, 5, 7),
  c(0.7, 25, 12, 15),
  c(0.85, 18, 2, 15),
  c(0.75, 22, 8, 10)
)
```

```r
for (i in 1:nrow(parValues))
{
  lines(1:100,
        generatePeristeraKostaki(par.c1 = parValues[i, 1], par.mu = parValues[i, 2],
                                 par.sigma = c(parValues[i, 3], parValues[i, 4])),
        col = i, lwd = grScale * 3)

  legend(60, 1 - 0.1 * (i - 1),
       legend = substitute(paste(c[1], " = ", pc1, ", ", mu, " = ", pmu, ", ",
                                 sigma[1], " = ", ps1, ", ", sigma[2], " = ", ps2),
                           list(pc1 = parValues[i, 1], pmu = parValues[i, 2],
                                ps1 = parValues[i, 3], ps2 = parValues[i, 4])),
       col = i,
       lwd = grScale * 3, cex = 0.8,
       title = NULL, bty = "n")
}

# 2. Mortality model (CoaleDemeny Life tables)

plot(c(1, 151), c(0, 1), type = "n",
     main = "Coale-Demeny Model Life Tables",
     xlab = "AGE\n",
     ylab = "q(x)",
     xlim = c(0, 100)
)

# five variations of parameter settings
parValues <- rbind(
  c("north", "M", 1),
  c("west", "F", 8),
  c("east", "M", 12),
  c("south", "F", 17),
  c("north", "F", 25)
)

for (i in 1:nrow(parValues))
{
  lines(1:151,
        generateCoaleDemenyLifeTable(region = parValues[i, 1],
                                     sex = parValues[i, 2],
                                     level = parValues[i, 3]),
        col = i, lwd = grScale * 3)

  legend(1, 1 - 0.1 * (i - 1),
       legend = paste("region = ", parValues[i, 1], ", ",
                      "sex = ", parValues[i, 2], ", ",
                      "level = ", parValues[i, 3]),
       col = i, text.font = 3,
       lwd = grScale * 3, cex = 0.8,
       title = NULL, bty = "n")
}

dev.off()
```

```
## pdf
##   2
```

```
knitr::include_graphics(plotName)
```