Monte Carlo simulation rainfall

2023-09-28

```
library(fitdistrplus)
library(pdqr)
library(truncdist)

# Load in required data
rainfall_data <- read.csv("rainfall_data.csv", header = T)

# Need to remove 0's for distribution fitting
rainfall_data[rainfall_data == 0] <- 0.001

# Determine monthly distribution for rainfall (EDA)
for (i in c(1:12)){
  descdist(rainfall_data[,(i+1)], discrete = FALSE)
}
```

```
#------------------#
# START ALGORITHM #
#------------------#

# READ IN DATA #
#--------------#
# Read in required rainfall data for central Texas (NOAA Station - WATSON)
rainfall_data <- read.csv("rainfall_data.csv", header = T)
names(rainfall_data) <- tolower(names(rainfall_data))
rainfall_data <- rainfall_data[,-1]

# GET PARAMETERS #
#----------------#
# Get simulation parameters
number_iterations <- as.numeric(readline("How many iterations of the simulation should we run?   "))
number_years <- as.numeric(readline("How many years should each iteration run?   "))

# Get rainfall distribution choice
dist_choice <- menu(choices = c("Normal", "Weibull", 'Exponential', "Custom"), title = "How should we mo
if(dist_choice == 1){
  dist_choice = 'norm'
} else if(dist_choice == 2){
  dist_choice = 'weibull'
} else if(dist_choice == 3){
  dist_choice == 'exp'
} else if(dist_choice == 4){
  dist_choice == 'custom'
}
```

```
# Expected min/max water usage per month
min_water <- as.numeric(readline("What is the minimum expected water usage per month?   "))
max_water <- as.numeric(readline("What is the maximum expected water usage per month?   "))

# Measurements
roof_area <- as.numeric(readline("What is the roof capture area (in square feet)?   "))
main_tank <- as.numeric(readline("What is the main tank volume (in gallons)?   "))
starting_level <- as.numeric(readline("What should the starting tank level be for each iteration?   "))

# Static efficency percentages
min_cap_eff <- 0.90
max_cap_eff <- 0.98
```

```
# FIT RAINFALL DISTRIBUTIONS #
#----------------------------#
# Normal Distribution requiring mean and standard deviation
if (dist_choice == 'norm'){

  rainfall_dists <- data.frame("month"=character(), "mean" = numeric(), "sd" = numeric(), "pval" = nume
  # Look through each month
  for(i in 1:12){
    # Fit normal distribution to month data
    x <- rainfall_data[,i]
    dist.x <- fitdist(x, 'norm')

    # Retrieve fitted data parameters
    mean <- as.numeric(dist.x$estimate[1])
    sd <- as.numeric(dist.x$estimate[2])

    # Calculate Kolmogorov-Smirnov Test of goodness of fit including associated p-value
    ks_res <- ks.test(x, 'pnorm', mean = mean, sd = sd)
    p_value <- ks_res$p.value

    # Add new month to existing data
    new_row <- data.frame("month" = colnames(rainfall_data[i]), "mean" = mean, "sd" = sd, "pval" = p_va
    rainfall_dists <- rbind(rainfall_dists, new_row)
  }

# Weibull Distribution requiring shape and scale
} else if (dist_choice == 'weibull'){

  rainfall_dists <- data.frame("month"=character(), "shape" = numeric(), "scale" = numeric(), "pval" = 
  # Look through each month
  for(i in 1:12){
    # Fit weibull distribution to month data
    x <- rainfall_data[,i]
    dist.x <- fitdist(x, 'weibull')

    # Retrieve fitted data parameters
    shape <- as.numeric(dist.x$estimate[1])
    scale <- as.numeric(dist.x$estimate[2])

    # Calculate Kolmogorov-Smirnov Test of goodness of fit including associated p-value
    ks_res <- ks.test(x, 'pweibull', scale= scale, shape = shape)
    p_value <- ks_res$p.value

    # Add new month to existing data
    new_row <- data.frame("month" = colnames(rainfall_data[i]), "shape" = shape, "scale" = scale, "pval
    rainfall_dists <- rbind(rainfall_dists, new_row)
  }

# Exponential Distribution requiring rate
} else if (dist_choice == 'exp'){

  rainfall_dists <- data.frame("month"=character(), "rate" = numeric(), "pval" = numeric())
  # Look through each month
  for(i in 1:12){
    # Fit exponential distribution to month data
    x <- rainfall_data[,i]
    dist.x <- fitdist(x, 'exp')

    # Retrieve fitted data parameters
    rate <- as.numeric(dist.x$estimate[1])

    # Calculate Kolmogorov-Smirnov Test of goodness of fit including associated p-value
    ks_res <- ks.test(x, 'pexp', rate =rate)
    p_value <- ks_res$p.value

    # Add new month to existing data
    new_row <- data.frame("month" = colnames(rainfall_data[i]), "rate" = rate, "pval" = p_value)
    rainfall_dists <- rbind(rainfall_dists, new_row)
  }

# Custom continuous random variable models
} else if (dist_choice == 'custom'){

  rainfall_dists <- list()
  for (i in 1:12){
    # Dynamically store each month's model to a list for future use
    new_dist <- new_r(rainfall_data[,i], type = 'continuous')
    rainfall_dists[[i]] <- new_dist
  }
}
```

```
# START MONTE CARLO SIMULATION #
#-------------------------------#
# Create tracking variables
iteration_stats <- list()
#iteration_stats <- data.frame('iteration'=as.numeric(), 'times_empty' = as.numeric(), 'times_overflow'

simulation_stats <- list()
#simulation_stats <- data.frame('iteration'=as.numeric(), 'year' = as.numeric(), 'month'=as.numeric(),
#                               'capture_perc'=as.numeric(), 'water_used' = as.numeric(), 'tank_before'

counter <- 1
for (i in 1:number_iterations){
```

```
# Create tracking variables
overflow <- 0
empty <- 0
current_amount <- starting_level

for (year in 1:number_years){

  for (month in 1:12){

    # Get current month rainfall amount in inches from our rainfall models
    rain_amount <- -1
    if(dist_choice == 'norm'){
      rain_amount = rtrunc(1, spec="norm", a=0, b=Inf,mean = rainfall_dists$mean[month], sd = rainfall
      #while(rain_amount < 0){
      #rain_amount = rnorm(1, mean = rainfall_dists$mean[month], sd = rainfall_dists$sd[month])
      #}
    } else if (dist_choice == 'weibull'){
      while(rain_amount < 0){
        rain_amount <- rweibull(1, shape = rainfall_dists$shape[month], scale = rainfall_dists$scale[m
      }
    } else if (dist_choice == 'exp'){
      while(rain_amount < 0){
        rain_amount <- rexp(1, rate = rainfall_dists$rate[month])
      }
    } else if(dist_choice == 'custom'){
      while(rain_amount < 0){
        rain_amount <- rainfall_dists[[month]](n=1)
      }
    }

    # Calculate total rainfall captured for month
    month_rain <- roof_area * (rain_amount / 12) * 7.48052
    # Calculate rainfall after capture inefficiencies
    perc_cap <- runif(1, min=min_cap_eff, max = max_cap_eff)
    month_rain <- month_rain * perc_cap

    # Calculate tradeoff with monthly usage
    month_usage <- runif(1, min = min_water, max = max_water)
    total_diff <- month_rain - month_usage

    # Calculate tank metrics
    previous_amount <- current_amount
    current_amount <- current_amount + total_diff

    # Error handle for overflow (dump excess to ground well) and empty
    spillage = 0
    if(current_amount > main_tank){
      overflow <- overflow + 1
      spillage <- current_amount - main_tank
      current_amount <- main_tank
    } else if(current_amount <= 0){
      empty <- empty + 1
      current_amount = 0
      }

      # Track metrics
      next_row <- data.frame('iteration'=i, 'year' = year, 'month'=month, 'rainfall_amount'=rain_amount
                             'capture_perc'=perc_cap, 'water_used' = month_usage, 'tank_before' = previ
                             'tank_after'=current_amount, 'overflow_amount' = spillage)
      simulation_stats[[counter]] <- next_row
      counter <- counter + 1
    }
  }

  # Track iteration stats
  new_row <- data.frame('iteration'=i, 'times_empty' = empty, 'times_overflow'=overflow)
  iteration_stats[[i]] <- new_row
}

# Bind rows together
simulation_stats <- bind_rows(simulation_stats)
iteration_stats <- bind_rows(iteration_stats)
```

```
# ANALYZE, OUTPUT, AND VISUALIZE RESULTS #
#----------------------------------------#

#Scneario 1: Increase storage tank capacity once

# recommended water tank storage increase = max spillage from original simulation + 20% tolerance
max(simulation_stats$spillage)
main_tank_increase <- main_tank + 1.2*max(simulation_stats$spillage)

# Create tracking variables
iteration_stats_tank_increase <- list()
#iteration_stats <- data.frame('iteration'=as.numeric(), 'times_empty' = as.numeric(), 'times_overflow'

simulation_stats_tank_increase <- list()
#simulation_stats <- data.frame('iteration'=as.numeric(), 'year' = as.numeric(), 'month'=as.numeric(),
#                               'capture_perc'=as.numeric(), 'water_used' = as.numeric(), 'tank_before'

counter <- 1
for (i in 1:number_iterations){

  # Create tracking variables
  overflow <- 0
  empty <- 0
  current_amount <- starting_level

  for (year in 1:number_years){

    for (month in 1:12){

      # Get current month rainfall amount in inches from our rainfall models
      rain_amount <- -1
      if(dist_choice == 'norm'){
        rain_amount = rtrunc(1, spec="norm", a=0, b=Inf,mean = rainfall_dists$mean[month], sd = rainfal
        #while(rain_amount < 0){
        #rain_amount = rnorm(1, mean = rainfall_dists$mean[month], sd = rainfall_dists$sd[month])
        #}
      } else if (dist_choice == 'weibull'){
        while(rain_amount < 0){
          rain_amount <- rweibull(1, shape = rainfall_dists$shape[month], scale = rainfall_dists$scale[
        }
      } else if (dist_choice == 'exp'){
        while(rain_amount < 0){
          rain_amount <- rexp(1, rate = rainfall_dists$rate[month])
        }
      } else if(dist_choice == 'custom'){
        while(rain_amount < 0){
          rain_amount <- rainfall_dists[[month]](n=1)
        }
      }

      # Calculate total rainfall captured for month
      month_rain <- roof_area * (rain_amount / 12) * 7.48052
      # Calculate rainfall after capture inefficencies
      perc_cap <- runif(1, min=min_cap_eff, max = max_cap_eff)
      month_rain <- month_rain * perc_cap

      # Calculate tradeoff with monthly usage
      month_usage <- runif(1, min = min_water, max = max_water)
      total_diff <- month_rain - month_usage

      # Calculate tank metrics
      previous_amount <- current_amount
      current_amount <- current_amount + total_diff

      # Error handle for overflow (dump excess to ground well) and empty
      spillage = 0
      if(current_amount > main_tank_increase){
        overflow <- overflow + 1
        spillage <- current_amount - main_tank_increase
        current_amount <- main_tank_increase
      } else if(current_amount <= 0){
        empty <- empty + 1
        current_amount = 0
      }

      # Track metrics
      next_row <- data.frame('iteration'=i, 'year' = year, 'month'=month, 'rainfall_amount'=rain_amount
                             'capture_perc'=perc_cap, 'water_used' = month_usage, 'tank_before' = previ
                             'tank_after'=current_amount, 'overflow_amount' = spillage)
      simulation_stats_tank_increase[[counter]] <- next_row
      counter <- counter + 1
    }
  }

  # Track iteration stats
  new_row <- data.frame('iteration'=i, 'times_empty' = empty, 'times_overflow'=overflow)
```

```
  iteration_stats_tank_increase[[i]] <- new_row
}

# Bind rows together
simulation_stats_tank_increase <- bind_rows(simulation_stats_tank_increase)
iteration_stats_tank_increase <- bind_rows(iteration_stats_tank_increase)

# Scenario 2: Increase storage tank capacity again for a 2nd time

# recommended water tank storage increase = max spillage from previous simulation + 20% tolerance
max(simulation_stats_tank_increase$spillage)
main_tank_increase2 <- main_tank_increase + 1.2*max(simulation_stats_tank_increase$spillage)

# Create tracking variables
iteration_stats_tank_increase2 <- list()
#iteration_stats <- data.frame('iteration'=as.numeric(), 'times_empty' = as.numeric(), 'times_overflow'

simulation_stats_tank_increase2 <- list()
#simulation_stats <- data.frame('iteration'=as.numeric(), 'year' = as.numeric(), 'month'=as.numeric(),
#                               'capture_perc'=as.numeric(), 'water_used' = as.numeric(), 'tank_before'

counter <- 1
for (i in 1:number_iterations){

  # Create tracking variables
  overflow <- 0
  empty <- 0
  current_amount <- starting_level

  for (year in 1:number_years){

    for (month in 1:12){

      # Get current month rainfall amount in inches from our rainfall models
      rain_amount <- -1
      if(dist_choice == 'norm'){
        rain_amount = rtrunc(1, spec="norm", a=0, b=Inf,mean = rainfall_dists$mean[month], sd = rainfal
        #while(rain_amount < 0){
        #rain_amount = rnorm(1, mean = rainfall_dists$mean[month], sd = rainfall_dists$sd[month])
        #}
      } else if (dist_choice == 'weibull'){
        while(rain_amount < 0){
          rain_amount <- rweibull(1, shape = rainfall_dists$shape[month], scale = rainfall_dists$scale[
        }
      } else if (dist_choice == 'exp'){
        while(rain_amount < 0){
          rain_amount <- rexp(1, rate = rainfall_dists$rate[month])
        }
      } else if(dist_choice == 'custom'){
        while(rain_amount < 0){
          rain_amount <- rainfall_dists[[month]](n=1)
        }
      }
```

```r
      # Calculate total rainfall captured for month
      month_rain <- roof_area * (rain_amount / 12) * 7.48052
      # Calculate rainfall after capture ineffcicencies
      perc_cap <- runif(1, min=min_cap_eff, max = max_cap_eff)
      month_rain <- month_rain * perc_cap

      # Calculate tradeoff with monthly usage
      month_usage <- runif(1, min = min_water, max = max_water)
      total_diff <- month_rain - month_usage

      # Calculate tank metrics
      previous_amount <- current_amount
      current_amount <- current_amount + total_diff

      # Error handle for overflow (dump excess to ground well) and empty
      spillage = 0
      if(current_amount > main_tank_increase2){
        overflow <- overflow + 1
        spillage <- current_amount - main_tank_increase2
        current_amount <- main_tank_increase2
      } else if(current_amount <= 0){
        empty <- empty + 1
        current_amount = 0
      }

      # Track metrics
      next_row <- data.frame('iteration'=i, 'year' = year, 'month'=month, 'rainfall_amount'=rain_amount
                             'capture_perc'=perc_cap, 'water_used' = month_usage, 'tank_before' = previ
                             'tank_after'=current_amount, 'overflow_amount' = spillage)
      simulation_stats_tank_increase2[[counter]] <- next_row
      counter <- counter + 1
    }
  }

  # Track iteration stats
  new_row <- data.frame('iteration'=i, 'times_empty' = empty, 'times_overflow'=overflow)
  iteration_stats_tank_increase2[[i]] <- new_row
}

# Bind rows together
simulation_stats_tank_increase2 <- bind_rows(simulation_stats_tank_increase2)
iteration_stats_tank_increase2 <- bind_rows(iteration_stats_tank_increase2)

# Scenario 3: Roof increase to 1.5 times original area, no storage tank increase
# Create tracking variables
iteration_stats_roof_increase <- list()
#iteration_stats <- data.frame('iteration'=as.numeric(), 'times_empty' = as.numeric(), 'times_overflow'=

simulation_stats_roof_increase <- list()
#simulation_stats <- data.frame('iteration'=as.numeric(), 'year' = as.numeric(), 'month'=as.numeric(),
#                               'capture_perc'=as.numeric(), 'water_used' = as.numeric(), 'tank_before'

counter <- 1
```

```r
for (i in 1:number_iterations){

  # Create tracking variables
  overflow <- 0
  empty <- 0
  current_amount <- starting_level

  for (year in 1:number_years){

    for (month in 1:12){

      # Get current month rainfall amount in inches from our rainfall models
      rain_amount <- -1
      if(dist_choice == 'norm'){
        rain_amount = rtrunc(1, spec="norm", a=0, b=Inf,mean = rainfall_dists$mean[month], sd = rainfall
        #while(rain_amount < 0){
        #rain_amount = rnorm(1, mean = rainfall_dists$mean[month], sd = rainfall_dists$sd[month])
        #}
      } else if (dist_choice == 'weibull'){
        while(rain_amount < 0){
          rain_amount <- rweibull(1, shape = rainfall_dists$shape[month], scale = rainfall_dists$scale[
        }
      } else if (dist_choice == 'exp'){
        while(rain_amount < 0){
          rain_amount <- rexp(1, rate = rainfall_dists$rate[month])
        }
      } else if(dist_choice == 'custom'){
        while(rain_amount < 0){
          rain_amount <- rainfall_dists[[month]](n=1)
        }
      }

      # Calculate total rainfall captured for month
      month_rain <- 1.5*roof_area * (rain_amount / 12) * 7.48052
      # Calculate rainfall after capture ineffecencies
      perc_cap <- runif(1, min=min_cap_eff, max = max_cap_eff)
      month_rain <- month_rain * perc_cap

      # Calculate tradeoff with monthly usage
      month_usage <- runif(1, min = min_water, max = max_water)
      total_diff <- month_rain - month_usage

      # Calculate tank metrics
      previous_amount <- current_amount
      current_amount <- current_amount + total_diff

      # Error handle for overflow (dump excess to ground well) and empty
      spillage = 0
      if(current_amount > main_tank){
        overflow <- overflow + 1
        spillage <- current_amount - main_tank
        current_amount <- main_tank
      } else if(current_amount <= 0){
        empty <- empty + 1
        current_amount = 0
      }

      # Track metrics
      next_row <- data.frame('iteration'=i, 'year' = year, 'month'=month, 'rainfall_amount'=rain_amount
                             'capture_perc'=perc_cap, 'water_used' = month_usage, 'tank_before' = previ
                             'tank_after'=current_amount, 'overflow_amount' = spillage)
      simulation_stats_roof_increase[[counter]] <- next_row
      counter <- counter + 1
    }
  }

  # Track iteration stats
  new_row <- data.frame('iteration'=i, 'times_empty' = empty, 'times_overflow'=overflow)
  iteration_stats_roof_increase[[i]] <- new_row
}

# Bind rows together
simulation_stats_roof_increase <- bind_rows(simulation_stats_roof_increase)
iteration_stats_roof_increase <- bind_rows(iteration_stats_roof_increase)

# Scenario 4: Roof increase to 1.5 times original area, combined with storage tank capacity increase in

# Create tracking variables
iteration_stats_roof_tank_increase <- list()
#iteration_stats <- data.frame('iteration'=as.numeric(), 'times_empty' = as.numeric(), 'times_overflow'=

simulation_stats_roof_tank_increase <- list()
#simulation_stats <- data.frame('iteration'=as.numeric(), 'year' = as.numeric(), 'month'=as.numeric(),
#                               'capture_perc'=as.numeric(), 'water_used' = as.numeric(), 'tank_before'

counter <- 1
for (i in 1:number_iterations){

  # Create tracking variables
  overflow <- 0
  empty <- 0
  current_amount <- starting_level

  for (year in 1:number_years){

    for (month in 1:12){

      # Get current month rainfall amount in inches from our rainfall models
      rain_amount <- -1
      if(dist_choice == 'norm'){
        rain_amount = rtrunc(1, spec="norm", a=0, b=Inf,mean = rainfall_dists$mean[month], sd = rainfal
        #while(rain_amount < 0){
        #rain_amount = rnorm(1, mean = rainfall_dists$mean[month], sd = rainfall_dists$sd[month])
        #}
      } else if (dist_choice == 'weibull'){
        while(rain_amount < 0){
          rain_amount <- rweibull(1, shape = rainfall_dists$shape[month], scale = rainfall_dists$scale[
        }
      } else if (dist_choice == 'exp'){
        while(rain_amount < 0){
          rain_amount <- rexp(1, rate = rainfall_dists$rate[month])
        }
      } else if(dist_choice == 'custom'){
        while(rain_amount < 0){
          rain_amount <- rainfall_dists[[month]](n=1)
        }
      }

      # Calculate total rainfall captured for month
      month_rain <- 1.5*roof_area * (rain_amount / 12) * 7.48052
      # Calculate rainfall after capture inefficencies
      perc_cap <- runif(1, min=min_cap_eff, max = max_cap_eff)
      month_rain <- month_rain * perc_cap

      # Calculate tradeoff with monthly usage
      month_usage <- runif(1, min = min_water, max = max_water)
      total_diff <- month_rain - month_usage

      # Calculate tank metrics
      previous_amount <- current_amount
      current_amount <- current_amount + total_diff

      # Error handle for overflow (dump excess to ground well) and empty
      spillage = 0
      if(current_amount > main_tank_increase){
        overflow <- overflow + 1
        spillage <- current_amount - main_tank_increase
        current_amount <- main_tank_increase
      } else if(current_amount <= 0){
        empty <- empty + 1
        current_amount = 0
      }

      # Track metrics
      next_row <- data.frame('iteration'=i, 'year' = year, 'month'=month, 'rainfall_amount'=rain_amount
                             'capture_perc'=perc_cap, 'water_used' = month_usage, 'tank_before' = previ
                             'tank_after'=current_amount, 'overflow_amount' = spillage)
      simulation_stats_roof_tank_increase[[counter]] <- next_row
      counter <- counter + 1
    }
  }

  # Track iteration stats
  new_row <- data.frame('iteration'=i, 'times_empty' = empty, 'times_overflow'=overflow)
  iteration_stats_roof_tank_increase[[i]] <- new_row
}

# Bind rows together
simulation_stats_roof_tank_increase <- bind_rows(simulation_stats_roof_tank_increase)
```

```r
iteration_stats_roof_tank_increase <- bind_rows(iteration_stats_roof_tank_increase)


# Scenario 5: Roof increase to 1.5 times original area, combined with storage tank capacity increase in
# Create tracking variables
iteration_stats_roof_tank_increase2 <- list()
#iteration_stats <- data.frame('iteration'=as.numeric(), 'times_empty' = as.numeric(), 'times_overflow'

simulation_stats_roof_tank_increase2 <- list()
#simulation_stats <- data.frame('iteration'=as.numeric(), 'year' = as.numeric(), 'month'=as.numeric(),
#                               'capture_perc'=as.numeric(), 'water_used' = as.numeric(), 'tank_before'

counter <- 1
for (i in 1:number_iterations){

  # Create tracking variables
  overflow <- 0
  empty <- 0
  current_amount <- starting_level

  for (year in 1:number_years){

    for (month in 1:12){

      # Get current month rainfall amount in inches from our rainfall models
      rain_amount <- -1
      if(dist_choice == 'norm'){
        rain_amount = rtrunc(1, spec="norm", a=0, b=Inf,mean = rainfall_dists$mean[month], sd = rainfal
        #while(rain_amount < 0){
        #rain_amount = rnorm(1, mean = rainfall_dists$mean[month], sd = rainfall_dists$sd[month])
        #}
      } else if (dist_choice == 'weibull'){
        while(rain_amount < 0){
          rain_amount <- rweibull(1, shape = rainfall_dists$shape[month], scale = rainfall_dists$scale[
        }
      } else if (dist_choice == 'exp'){
        while(rain_amount < 0){
          rain_amount <- rexp(1, rate = rainfall_dists$rate[month])
        }
      } else if(dist_choice == 'custom'){
        while(rain_amount < 0){
          rain_amount <- rainfall_dists[[month]](n=1)
        }
      }

      # Calculate total rainfall captured for month
      month_rain <- 1.5*roof_area * (rain_amount / 12) * 7.48052
      # Calculate rainfall after capture ineffciencies
      perc_cap <- runif(1, min=min_cap_eff, max = max_cap_eff)
      month_rain <- month_rain * perc_cap

      # Calculate tradeoff with monthly usage
      month_usage <- runif(1, min = min_water, max = max_water)
      total_diff <- month_rain - month_usage

      # Calculate tank metrics
      previous_amount <- current_amount
      current_amount <- current_amount + total_diff

      # Error handle for overflow (dump excess to ground well) and empty
      spillage = 0
      if(current_amount > main_tank_increase2){
        overflow <- overflow + 1
        spillage <- current_amount - main_tank_increase2
        current_amount <- main_tank_increase2
      } else if(current_amount <= 0){
        empty <- empty + 1
        current_amount = 0
      }

      # Track metrics
      next_row <- data.frame('iteration'=i, 'year' = year, 'month'=month, 'rainfall_amount'=rain_amount
                             'capture_perc'=perc_cap, 'water_used' = month_usage, 'tank_before' = previ
                             'tank_after'=current_amount, 'overflow_amount' = spillage)
      simulation_stats_roof_tank_increase2[[counter]] <- next_row
      counter <- counter + 1
    }
  }

  # Track iteration stats
  new_row <- data.frame('iteration'=i, 'times_empty' = empty, 'times_overflow'=overflow)
  iteration_stats_roof_tank_increase2[[i]] <- new_row
}

# Bind rows together
simulation_stats_roof_tank_increase2 <- bind_rows(simulation_stats_roof_tank_increase2)
iteration_stats_roof_tank_increase2 <- bind_rows(iteration_stats_roof_tank_increase2)

# Compare key metrics of all scenarios
sum(iteration_stats$times_empty)
sum(iteration_stats_tank_increase$times_empty)
sum(iteration_stats_tank_increase2$times_empty)
sum(iteration_stats_roof_increase$times_empty)
sum(iteration_stats_roof_tank_increase$times_empty)
sum(iteration_stats_roof_tank_increase2$times_empty)

sum(iteration_stats$times_overflow)
sum(iteration_stats_tank_increase$times_overflow)
sum(iteration_stats_tank_increase2$times_overflow)
sum(iteration_stats_roof_increase$times_overflow)
sum(iteration_stats_roof_tank_increase$times_overflow)
sum(iteration_stats_roof_tank_increase2$times_overflow)
```

```
# START Climate Change MONTE CARLO SIMULATION #
#------------------------------#
# Create tracking variables
```

```r
iteration_stats_2pctSD <- list()
#iteration_stats <- data.frame('iteration'=as.numeric(), 'times_empty' = as.numeric(), 'times_overflow'

simulation_stats_2pctSD <- list()
#simulation_stats <- data.frame('iteration'=as.numeric(), 'year' = as.numeric(), 'month'=as.numeric(),
#                                'capture_perc'=as.numeric(), 'water_used' = as.numeric(), 'tank_before'

counter <- 1
for (i in 1:number_iterations){

  # Update progress Bar
  #setWinProgressBar(progressBar, i, label = paste("Running Iteration ", i, sep = ""))

  # Create tracking variables
  overflow <- 0
  empty <- 0
  current_amount <- starting_level

  for (year in 1:number_years){

    for (month in 1:12){

      # Get current month rainfall amount in inches from our rainfall models
      rain_amount <- -1
      if(dist_choice == 'norm'){
        rain_amount = rtrunc(1, spec="norm", a=0, b=Inf,mean = rainfall_dists$mean[month], sd = rainfal
        #while(rain_amount < 0){
          #rain_amount = rnorm(1, mean = rainfall_dists$mean[month], sd = rainfall_dists$sd[month])
        #}
      } else if (dist_choice == 'weibull'){
        while(rain_amount < 0){
          rain_amount <- rweibull(1, shape = rainfall_dists$shape[month], scale = rainfall_dists$scale[
        }
      } else if (dist_choice == 'exp'){
        while(rain_amount < 0){
          rain_amount <- rexp(1, rate = rainfall_dists$rate[month])
        }
      } else if(dist_choice == 'custom'){
        while(rain_amount < 0){
          rain_amount <- rainfall_dists[[month]](n=1)
        }
      }

      # Calculate total rainfall captured for month
      month_rain <- roof_area * (rain_amount / 12) * 7.48052
      # Calculate rainfall after capture inefficencies
      perc_cap <- runif(1, min=min_cap_eff, max = max_cap_eff)
      month_rain <- month_rain * perc_cap

      # Calculate tradeoff with monthly usage
      month_usage <- runif(1, min = min_water, max = max_water)
      total_diff <- month_rain - month_usage
      # Calculate tank metrics
      previous_amount <- current_amount
      current_amount <- current_amount + total_diff

      # Error handle for overflow (dump excess to ground well) and empty
      if(current_amount > main_tank){
        overflow <- overflow + 1
        current_amount <- main_tank
      } else if(current_amount <= 0){
        empty <- empty + 1
        current_amount = 0
      }

      # Track metrics
      next_row <- data.frame('iteration'=i, 'year' = year, 'month'=month, 'rainfall_amount'=rain_amount
                             'capture_perc'=perc_cap, 'water_used' = month_usage, 'tank_before' = previ
      simulation_stats_2pctSD[[counter]] <- next_row
      counter <- counter + 1
      #simulation_stats <- rbind(simulation_stats, next_row)
    }
  }

  # Track iteration stats
  new_row <- data.frame('iteration'=i, 'times_empty' = empty, 'times_overflow'=overflow)
  iteration_stats_2pctSD[[i]] <- new_row
}

# Bind rows together
simulation_stats_2pctSD <- bind_rows(simulation_stats_2pctSD)
iteration_stats_2pctSD <- bind_rows(iteration_stats_2pctSD)


# Graphs of key metrics

# Compare rainfall distribution
par(mfrow = c(1,2))
boxplot(simulation_stats$rainfall_amount, ylim = c(0,30), ylab = "Rainfall", main = "Boxplot of Origina
boxplot(simulation_stats_2pctSd$rainfall_amount, ylim = c(0,30), ylab = "Rainfall", main = "Boxplot of 2
par(mfrow = c(1,1))

# Compare frequency of empty tank
par(mfrow = c(1,2))
hist(iteration_stats$times_empty,col = "darkblue", xlab = "No. of times tank is empty", ylim = c(0, 100
     main = "Histogram of Original sim")
hist(iteration_stats_2pctSD$times_empty, col = "darkblue", xlab = "No. of times tank is empty", ylim = c
     main = "Histogram of 2% SD annual rate increase")
par(mfrow = c(1,1))

# Comprae ferquency of tank overflow
par(mfrow = c(1,2))
hist(iteration_stats$times_overflow,col = "darkgreen", xlab = "No. of times tank overflows", ylim = c(0
     main = "Histogram of Original sim")
hist(iteration_stats_2pctSD$times_overflow, col = "darkgreen", xlab = "No. of times tank overflows", yli
     main = "Histogram of 2% SD annual rate increase")
par(mfrow = c(1,1))
```