```
OICD_server <- function(input,output,session) {

# set to defaut risky ------------------------------------------------------------

  observeEvent(input$setToDefault_Parameter_x,{
    updateSliderInput(session, inputId= "capital_x",
                      label="capital",
                      min = 1,
                      max = 100,
                      step = 1,
                      value = 20)
    updateSliderInput(session, inputId = "price_x",
                      label= "price",
                      min = 1,
                      max = 100,
                      step = 1,
                      value = 19)
    updateSliderInput(session, inputId = "beta_x",
                      label= "discount factor",
                      min = 0,
                      max = 1,
                      step = 0.01,
                      value = 0.9)
    updateSliderInput(session, inputId = "gamma_x",
                      label= "constant relative risk aversion",
                      min = 0.01,
                      max = 5,
                      step = 0.01,
                      value = 2)
    updateSliderInput(session, inputId = "delta_x",
                      label= "isoquant delta",
                      min = -0.8,
                      max = 0.8,
                      step = 0.1,
                      value = 0)
  })

  observeEvent(input$setToDefault_investment_x,{
    updateSliderInput(session, inputId = "probability1_x",
                      label= "probability1",
                      min = 0,
                      max = 1,
                      step = 0.01,
                      value = 0.2)
    updateSliderInput(session, inputId= "dividend1_x",
                      label="dividend1",
                      min = 1,
                      max = 100,
                      step= 1,
                      value = 5)
    updateSliderInput(session, inputId = "probability2_x",
                      label= "probability2",
                      min = 0,
                      max = 1,
                      step = 0.01,
                      value = 0.5)
    updateSliderInput(session, inputId= "dividend2_x",
                      label="dividend2",
                      min = 1,
                      max = 100,
```

```
                      step= 1,
                      value = 15)

    updateSliderInput(session, inputId = "probability3_x",
                      label= "probability3",
                      min = 0,
                      max = 1,
                      step = 0.01,
                      value = 0.3)

    updateSliderInput(session, inputId= "dividend3_x",
                      label="dividend3",
                      min = 1,
                      max = 100,
                      step= 1,
                      value = 25)
  })


# adjust probabilities risky ----------------------------------------------------

  observeEvent(input$probability1_x,{
    if(input$probability1_x + input$probability2_x + input$probability3_x > 1){
      if(input$probability1_x + input$probability2_x > 1){
        updateSliderInput(session, inputId = "probability3_x",
                      value = 0)
        updateSliderInput(session, inputId = "probability2_x",
                      value = 1 - input$probability1_x)}
      else if(input$probability1_x + input$probability2_x < 1){
        updateSliderInput(session, inputId = "probability3_x",
                      value = 1 - input$probability1_x - input$probability2_x)}}
    else if(input$probability1_x + input$probability2_x + input$probability3_x <
1){
        updateSliderInput(session, inputId = "probability3_x",
                      value = 1 - input$probability1_x - input$probability2_x)}
  })

  observeEvent(input$probability2_x,{
    if(input$probability1_x + input$probability2_x + input$probability3_x > 1){
      if(input$probability1_x + input$probability2_x > 1){
        updateSliderInput(session, inputId = "probability3_x",
                      value = 0)
        updateSliderInput(session, inputId = "probability1_x",
                      value = 1 - input$probability2_x)}
      else if(input$probability1_x + input$probability2_x < 1){
        updateSliderInput(session, inputId = "probability3_x",
                      value = 1 - input$probability1_x - input$probability2_x)}}
    else if(input$probability1_x + input$probability2_x + input$probability3_x <
1){
      updateSliderInput(session, inputId = "probability3_x",
                      value = 1 - input$probability1_x - input$probability2_x)}
  })

  observeEvent(input$probability3_x,{
    if(input$probability1_x + input$probability2_x + input$probability3_x > 1){
      if(input$probability1_x + input$probability3_x > 1){
        updateSliderInput(session, inputId = "probability2_x",
                      value = 0)
        updateSliderInput(session, inputId = "probability1_x",
                      value = 1 - input$probability3_x)}
```

```
      else if(input$probability1_x + input$probability3_x < 1){
        updateSliderInput(session, inputId = "probability2_x",
                          value = 1 - input$probability1_x - input$probability3_x)}}
    else if(input$probability1_x + input$probability2_x + input$probability3_x <
1){
      updateSliderInput(session, inputId = "probability2_x",
                        value = 1 - input$probability1_x - input$probability3_x)}
  })


# import parameters risky ----------------------------------------------------

  parameters_x <- reactive({
    par <- list()
    par$capital <- input$capital_x
    par$price <- input$price_x
    par$beta <- input$beta_x
    par$gamma <- input$gamma_x
    par$delta <- input$delta_x
    return(par)
  })

  probabilities_x <- reactive({
    probs <- c(input$probability1_x, input$probability2_x, input$probability3_x)
    return(probs)
  })

  dividends_x <- reactive({
    divs <- c(input$dividend1_x, input$dividend2_x, input$dividend3_x)
    return(divs)
  })

  x_axis_x <- reactive({
    scale_x <- 200
    return(seq(1/scale_x, (scale_x-1)/scale_x, 1/scale_x))})


# find maximal aggregated utility for given parameters risky -------------------

  MaxExpTotUtility_x <- reactive({
    max_x.l <- list()
    par_x.l <- parameters_x()
    pi.v <- probabilities_x()
    d1.v <- dividends_x()
    x.v <- par_x.l$capital/par_x.l$price*x_axis_x()
    x.d <- c(min(x.v), max(x.v))

    u.f <- function(c, gamma = par_x.l$gamma){
      if(gamma == 1){return(log(c))}
      else{return((c^(1-gamma)-1)/(1-gamma))}}

    bE_u1.f <- function(x, p0 = par_x.l$price, beta = par_x.l$beta, pi = pi.v,
                        d1 = d1.v){
      return(beta*(pi[1]*u.f(x*d1[1])+pi[2]*u.f(x*d1[2])+pi[3]*u.f(x*d1[3])))}

    U_x.f <- function(x, K=par_x.l$capital, p0 = par_x.l$price){
      return(u.f(K - x * p0) + bE_u1.f(x))}

    f.v <- U_x.f(x.v)
    max_x.l$objective <- max(f.v, na.rm = TRUE)
```

```
    max_x.l$maximum <- x.v[which.max(f.v)]

    return(max_x.l)
  })



# portfolio choice plot risky ------------------------------------------------

  output$PortfolioChoice_x <- renderPlot({
    max_x.l <- MaxExpTotUtility_x()
    par_x.l <- parameters_x()
    pi.v <- probabilities_x()
    d1.v <- dividends_x()

    x.v <- par_x.l$capital*x_axis_x()


    u.f <- function(c, gamma = par_x.l$gamma){
      if(gamma == 1){return(log(c))}
      else{return((c^(1-gamma)-1)/(1-gamma))}}

    bE_u1.f <- function(xp0, p0 = par_x.l$price, beta = par_x.l$beta, pi = pi.v,
                        d1 = d1.v){
      x <- xp0/p0
      return(beta*(pi[1]*u.f(x*d1[1])+pi[2]*u.f(x*d1[2])+pi[3]*u.f(x*d1[3])))}

    U_x.f <- function(xp0, K=par_x.l$capital, p0=par_x.l$price){
      return(u.f(K - xp0) + bE_u1.f(xp0))}



    U_x.v <- U_x.f(x.v)
    u.v <- u.f(par_x.l$capital - x.v)
    bE_u1.v <- mapply(bE_u1.f, x.v)




    fmax <- max(u.f(par_x.l$capital - par_x.l$price*max_x.l$maximum),
                bE_u1.f(par_x.l$price*max_x.l$maximum),
                U_x.f(par_x.l$price*max_x.l$maximum))
    fmin <- min(u.f(par_x.l$capital - par_x.l$price*max_x.l$maximum),
                bE_u1.f(par_x.l$price*max_x.l$maximum),
                U_x.f(par_x.l$price*max_x.l$maximum))


    xmax <- max(x.v)
    xmin <- min(x.v)
    ymax <- fmax + (fmax - fmin)
    ymin <- fmin - (fmax - fmin)


    plot(NA,NA,xaxs="r",yaxs="r",xlim=c(xmin,xmax), ylim=c(ymin, ymax),
         main="Portfolio Choice", xlab="xp0", ylab="U(c0,c1), u(c0), bE[u(c1)]",
         yaxt='n')

    legend("topright", c("U(xp0)", "u0", "beta*E[u1]"),
           fill = c("red", "blue", "green"))

```

```
    lines(x.v, U_x.v, col="red")
    lines(x.v, u.v, col = "blue")
    lines(x.v, bE_u1.v, col = "green")

    points(par_x.l$price*max_x.l$maximum, max_x.l$objective)

  })



# isoquant plot risky ------------------------------------------------------------

  output$Isoquants_x <- renderPlot({
    max_x.l <- MaxExpTotUtility_x()
    par_x.l <- parameters_x()
    pi.v <- probabilities_x()
    d1.v <- dividends_x()

    x.v <- par_x.l$capital*x_axis_x()

    maximum <- max_x.l$maximum
    f_max <- max_x.l$objective

    u.f <- function(c, gamma = par_x.l$gamma){
      if(gamma == 1){return(log(c))}
      else{return((c^(1-gamma)-1)/(1-gamma))}}

    bE_u1.f <- function(x, p0 = par_x.l$price, beta = par_x.l$beta, pi = pi.v,
                        d1 = d1.v){
      return(beta*(pi[1]*u.f(x/p0*d1[1])+pi[2]*u.f(x/p0*d1[2])+
                     pi[3]*u.f(x/p0*d1[3])))}

    U_cx.f <- function(c, xp0, K=par_x.l$capital, p0=par_x.l$price){
      return(u.f(c) + bE_u1.f(xp0))}

    KK <- function(xp0, K = par_x.l$capital){return(K - xp0)}

    cons_x.f <- function(xp0, K=par_x.l$capital, p0=par_x.l$price,
          beta=par_x.l$beta, gamma=par_x.l$gamma, f_const = max_x.l$objective){
      if(gamma == 1){c0 <- exp(f_const - bE_u1.f(xp0))}
      else{c0 <- ((f_const - bE_u1.f(xp0))*(1-gamma)+1)^(1/(1-gamma))}
      return(c0)}

    xmax <- 1.2*par_x.l$capital
    xmin <- 0
    ymax <- 1.2*par_x.l$capital
    ymin <- 0
    plot(NA,NA,xaxs="r",yaxs="r",xlim=c(xmin,xmax), ylim=c(ymin, ymax),
         main="ISOQUANTS", xlab="xp0", ylab="c0")

    lines(x.v, KK(x.v))



    # Isoquant for U_max

    f_c <- U_cx.f(c = (par_x.l$capital - max_x.l$maximum*par_x.l$price),
                  xp0 = max_x.l$maximum*par_x.l$price, par_x.l$capital)
    x.v2 <- x.v
    cons_x.v  <- cons_x.f(x.v, f_const=f_c)
```

```
    cons_x.v[1: which.max(cons_x.v)] <- NA
    cons_x.v[cons_x.v > par_x.l$capital] <- NA

    k <- which.min(abs(cons_x.v - par_x.l$capital))

    cons_x.v[k - 1] <- par_x.l$capital
    x.v2[k-1] <- x.v2[k] + (x.v2[k] - x.v2[k+1])/(cons_x.v[k]
                                - cons_x.v[k+1])*(cons_x.v[k-1] - cons_x.v[k])
    lines(x.v2, cons_x.v)


    # Isoquant for U(delta)

    f_c <- U_cx.f(
      c = (par_x.l$capital-max_x.l$maximum*par_x.l$price)*(1+par_x.l$delta),
                  xp0 = max_x.l$maximum*par_x.l$price*(1+par_x.l$delta))
    x.v3 <- x.v
    cons_x.v  <- cons_x.f(x.v, f_const=f_c)
    cons_x.v[1: which.max(cons_x.v)] <- NA
    cons_x.v[cons_x.v > par_x.l$capital] <- NA
    k <- which.min(abs(cons_x.v - par_x.l$capital))

    cons_x.v[k - 1] <- par_x.l$capital
    x.v3[k-1] <- x.v2[k] + (x.v2[k] - x.v2[k+1])/(cons_x.v[k]
                                  - cons_x.v[k+1])*(cons_x.v[k-1] - cons_x.v[k])
    lines(x.v3, cons_x.v)




    points(max_x.l$maximum*par_x.l$price,
           par_x.l$capital - max_x.l$maximum*par_x.l$price)
    legend("topright", c(c("xp0* = ", round(max_x.l$maximum*par_x.l$price, 2)),
              c("c0* = ",
                round(par_x.l$capital - max_x.l$maximum*par_x.l$price, 2))))

  })



# marginal utility plot risky ----------------------------------------------------

  output$MarginalUtility_x <- renderPlot({
    max_x.l <- MaxExpTotUtility_x()
    par_x.l <- parameters_x()
    pi.v <- probabilities_x()
    d1.v <- dividends_x()

    x.v <- par_x.l$capital*x_axis_x()


   u.f <- function(c, gamma = par_x.l$gamma){
      if(gamma == 1){return(log(c))}
      else{return((c^(1-gamma)-1)/(1-gamma))}}

    bE_u1.f <- function(x, p0 = par_x.l$price, beta = par_x.l$beta, pi = pi.v,
                        d1 = d1.v){
      return(beta*(pi[1]*u.f(x/p0*d1[1])+pi[2]*u.f(x/p0*d1[2])+
                     pi[3]*u.f(x/p0*d1[3])))}

```

```
U_cx.f <- function(c, xp0, K=par_x.l$capital, p0=par_x.l$price){
  return(u.f(c) + bE_u1.f(xp0))}

U_c.f <- function(c, xp0 = max_x.l$maximum*par_x.l$price, K=par_x.l$capital,
                  p0=par_x.l$price){
  return(u.f(c) + bE_u1.f(xp0))}

U_x.f <- function(xp0, c = par_x.l$capital-max_x.l$maximum*par_x.l$price,
                  K=par_x.l$capital, p0=par_x.l$price){
  return(u.f(c) + bE_u1.f(xp0))}


xmax <- par_x.l$capital
xmin <- 0

# fit y_axis limits to the plot
p <- max_x.l$maximum*par_x.l$price
q <- par_x.l$capital - p
x1 <- max(xmin, q - (xmax-xmin)/3, na.rm = TRUE)
x2 <- min(xmax, q + (xmax-xmin)/3, na.rm = TRUE)
slope <- (U_c.f(q*1.01)-U_c.f(q*0.99))/(q*1.01 - q*0.99)
U_c.f1 <- U_c.f(q) + (x1-q)*slope
U_c.f2 <- U_c.f(q) + (x2-q)*slope
x1 <- max(xmin, p - (xmax-xmin)/3, na.rm = TRUE)
x2 <- min(xmax, p + (xmax-xmin)/3, na.rm = TRUE)
slope <- (U_x.f(p*1.01)-U_x.f(p*0.99))/(p*1.01 - p*0.99)
U_x.f1 <- U_x.f(p) + (x1-p)*slope
U_x.f2 <- U_x.f(p) + (x2-p)*slope
yaxismax <- max(U_x.f2, U_c.f2)
yaxismin <- min(U_x.f1, U_c.f1)
yaxisdist <- yaxismax - yaxismin

ymax <- yaxismax + 0.5*yaxisdist
ymin <- yaxismin - 0.5*yaxisdist

par(mfrow = c(1, 2))
plot(NA,NA,xaxs="r",yaxs="r",xlim=c(xmin,xmax), ylim=c(ymin, ymax),
     main="marginal utility c0", xlab="c0", ylab="", yaxt='n')

lines(x.v, U_c.f(x.v))


# tangent to immediate consumption part
tangent_U_c.f <- function(p){
  x1 <- max(xmin, p - (xmax-xmin)/3, na.rm = TRUE)
  x2 <- min(xmax, p + (xmax-xmin)/3, na.rm = TRUE)
  slope <- (U_c.f(p*1.01)-U_c.f(p*0.99))/(p*1.01 - p*0.99)
  U_c.f1 <- U_c.f(p) + (x1-p)*slope
  U_c.f2 <- U_c.f(p) + (x2-p)*slope
  lines(c(x1, x2), c(U_c.f1, U_c.f2))
  points(p, U_c.f(p))
  return(slope)}

s <- tangent_U_c.f(par_x.l$capital - max_x.l$maximum*par_x.l$price)
s <- round(s, abs(log10(s))+1)

legend("bottomright", c("dU/dc0 = ", s))
```

```
421
422
423
424     plot(NA,NA,xaxs="r",yaxs="r",xlim=c(xmin,xmax),
425          ylim=c(ymin, ymax), main="marginal utility beta*E[u(c1)]",
426          xlab="xp0", ylab="", yaxt='n')
427
428     lines(x.v, U_x.f(x.v))
429
430     # tangent to investment part
431     tangent_U_x.f <- function(p){
432       x1 <- max(xmin, p - (xmax-xmin)/3, na.rm = TRUE)
433       x2 <- min(xmax, p + (xmax-xmin)/3, na.rm = TRUE)
434       slope <- (U_x.f(p*1.01)-U_x.f(p*0.99))/(p*1.01 - p*0.99)
435       U_x.f1 <- U_x.f(p) + (x1-p)*slope
436       U_x.f2 <- U_x.f(p) + (x2-p)*slope
437       lines(c(x1, x2), c(U_x.f1, U_x.f2))
438       points(p, U_x.f(p))
439       return(slope)}
440
441     t <- tangent_U_x.f(max_x.l$maximum*par_x.l$price)
442     t <- round(t, abs(log10(t))+1)
443
444     legend("bottomright", c("dU/dc1 =", t))
445
446
447
448   })
449
450
451
452 # results output risky ------------------------------------------------------
453
454   output$maxZf_x <- renderText({
455     max_x.l <- MaxExpTotUtility_x()
456     paste("max. exp. Utility: ", round(max_x.l$objective, 3))
457   })
458
459   output$max_x <- renderText({
460     max_x.l <- MaxExpTotUtility_x()
461     paste("x*: ", round(max_x.l$maximum, 3))
462   })
463
464
465
466 # set to default riskless ---------------------------------------------------
467 -
468
469   observeEvent(input$setToDefault_Parameter_xy,{
470     updateSliderInput(session, inputId= "capital_xy",
471                       label="capital",
472                       min = 1,
473                       max = 100,
474                       step = 1,
475                       value = 20)
476     updateSliderInput(session, inputId = "price_p0_xy",
477                       label= "price risky investment",
478                       min = 1,
479                       max = 100,
480                       step = 1,
```

```
                      value = 19)
    updateSliderInput(session, inputId = "price_b0_xy",
                      label= "price riskless investment",
                      min = 0.05,
                      max = 5,
                      step = 0.05,
                      value = 0.95)
    updateSliderInput(session, inputId = "beta_xy",
                      label= "discount factor",
                      min = 0,
                      max = 1,
                      step = 0.01,
                      value = 0.9)
    updateSliderInput(session, inputId = "gamma_xy",
                      label= "constant relative risk aversion",
                      min = 0.01,
                      max = 5,
                      step = 0.01,
                      value = 2)
    updateSliderInput(session, inputId = "delta_xy",
                      label= "isoquant delta y",
                      min = -1,
                      max = 1,
                      step = 0.05,
                      value = 0)
  })

  observeEvent(input$setToDefault_investment_xy,{
    updateSliderInput(session, inputId = "probability1_xy",
                      label= "probability1",
                      min = 0,
                      max = 1,
                      step = 0.01,
                      value = 0.2)
    updateSliderInput(session, inputId= "dividend1_xy",
                      label="dividend1",
                      min = 1,
                      max = 100,
                      step= 1,
                      value = 5)
    updateSliderInput(session, inputId = "probability2_xy",
                      label= "probability2",
                      min = 0,
                      max = 1,
                      step = 0.01,
                      value = 0.5)
    updateSliderInput(session, inputId= "dividend2_xy",
                      label="dividend2",
                      min = 1,
                      max = 100,
                      step= 1,
                      value = 15)

    updateSliderInput(session, inputId = "probability3_xy",
                      label= "probability3",
                      min = 0,
                      max = 1,
                      step = 0.01,
                      value = 0.3)

```

```
    updateSliderInput(session, inputId= "dividend3_xy",
                      label="dividend3",
                      min = 1,
                      max = 100,
                      step= 1,
                      value = 25)
  })



# adjust probabilities riskless ------------------------------------------------

  observeEvent(input$probability1_xy,{
    if(input$probability1_xy+input$probability2_xy+input$probability3_xy > 1){
      if(input$probability1_xy + input$probability2_xy > 1){
        updateSliderInput(session, inputId = "probability3_xy", value = 0)
        updateSliderInput(session, inputId = "probability2_xy",
                    value = 1 - input$probability1_xy)}
      else if(input$probability1_xy + input$probability2_xy < 1){
        updateSliderInput(session, inputId = "probability3_xy",
                    value = 1 - input$probability1_xy - input$probability2_xy)}}
    else if(input$probability1_xy+input$probability2_xy+input$probability3_xy <
1){
      updateSliderInput(session, inputId = "probability3_xy",
                    value = 1 - input$probability1_xy - input$probability2_xy)}
  })

  observeEvent(input$probability2_xy,{
    if(input$probability1_xy+input$probability2_xy+input$probability3_xy > 1){
      if(input$probability1_xy + input$probability2_xy > 1){
        updateSliderInput(session, inputId = "probability3_xy", value = 0)
        updateSliderInput(session, inputId = "probability1_xy",
                    value = 1 - input$probability2_xy)}
      else if(input$probability1_xy + input$probability2_xy < 1){
        updateSliderInput(session, inputId = "probability3_xy",
                    value = 1 - input$probability1_xy - input$probability2_xy)}}
    else if(input$probability1_xy+input$probability2_xy+input$probability3_xy <
1){
      updateSliderInput(session, inputId = "probability3_xy",
                    value = 1 - input$probability1_xy - input$probability2_xy)}
  })

  observeEvent(input$probability3_xy,{
    if(input$probability1_xy+input$probability2_xy+input$probability3_xy > 1){
      if(input$probability1_xy + input$probability3_xy > 1){
        updateSliderInput(session, inputId = "probability2_xy", value = 0)
        updateSliderInput(session, inputId = "probability1_xy",
                    value = 1 - input$probability3_xy)}
      else if(input$probability1_xy + input$probability3_xy < 1){
        updateSliderInput(session, inputId = "probability2_xy",
                    value = 1 - input$probability1_xy - input$probability3_xy)}}
    else if(input$probability1_xy+input$probability2_xy+input$probability3_xy <
1){
      updateSliderInput(session, inputId = "probability2_xy",
                    value = 1 - input$probability1_xy - input$probability3_xy)}
  })



# import parameters riskless ---------------------------------------------------
```

```
  parameters_xy <- reactive({
    par <- list()
    par$capital <- input$capital_xy
    par$price <- input$price_p0_xy
    par$brice <- input$price_b0_xy
    par$beta <- input$beta_xy
    par$gamma <- input$gamma_xy
    par$delta <- input$delta_xy
    return(par)
  })

  probabilities_xy <- reactive({
    probs <- c(input$probability1_xy, input$probability2_xy,
               input$probability3_xy)
    return(probs)
  })

  dividends_xy <- reactive({
    divs <- c(input$dividend1_xy, input$dividend2_xy, input$dividend3_xy)
    return(divs)
  })

  x_axis_xy <- reactive({
    scale_xy <- 200
    return(seq(0/scale_xy, (scale_xy - 1)/scale_xy, 1/scale_xy))})



# find maximal aggregated utility for given parameters riskless ----------------

  MaxExpTotUtility_xy <- reactive({
    max_xy.l <- list()
    par_xy.l <- parameters_xy()
    pi.v <- probabilities_xy()
    d1.v <- dividends_xy()
    v.v <- par_xy.l$capital*x_axis_xy()
    v.d <- c(min(v.v), max(v.v))

    u.f <- function(c, gamma = par_xy.l$gamma){
      if(gamma == 1){return(log(c))}
      else{return((c^(1-gamma)-1)/(1-gamma))}}

    bE_u1.f <- function(xp0, yb0, p0=par_xy.l$price, b0=par_xy.l$brice, pi=pi.v,
                        d1=d1.v, beta=par_xy.l$beta){
      return(beta*(pi[1]*u.f(xp0/p0*d1[1]+yb0/b0))+
               (pi[2]*u.f(xp0/p0*d1[2]+yb0/b0))+
               (pi[3]*u.f(xp0/p0*d1[3]+yb0/b0)))}

    U_xp0yb0.f <- function(xp0, yb0, K=par_xy.l$capital, p0=par_xy.l$price,
                        b0=par_xy.l$brice){
      mapply(function(xp, yb){
        if(K - xp - yb < 0){return(NA)}
        else{return(u.f(K - xp - yb) + bE_u1.f(xp, yb))}}, xp = xp0, yb = yb0)}

    f.m <- outer(v.v, v.v, U_xp0yb0.f)

    max_xy.l$objective <- max(f.m, na.rm = TRUE)
    max_xy.l$maximum_x <- v.v[which.max(f.m)%%length(v.v)]
    max_xy.l$maximum_y <- v.v[which.max(f.m)%/%length(v.v)+1]
```

```
    return(max_xy.l)
  })



# portfolio choice plot riskless ------------------------------------------------

  output$PortfolioChoice_xy <- renderPlot({
    max_xy.l <- MaxExpTotUtility_xy()
    par_xy.l <- parameters_xy()
    pi.v <- probabilities_xy()
    d1.v <- dividends_xy()

    v.v <- par_xy.l$capital*x_axis_xy()
    v.d <- c(min(v.v), max(v.v))

    u.f <- function(c, gamma = par_xy.l$gamma){
      if(gamma == 1){return(log(c))}
      else{return((c^(1-gamma)-1)/(1-gamma))}}

    bE_u1.f <- function(xp0, yb0, p0=par_xy.l$price, b0=par_xy.l$brice, pi=pi.v,
                        d1=d1.v, beta=par_xy.l$beta){
      return(beta*(pi[1]*u.f(xp0/p0*d1[1]+yb0/b0))+
               (pi[2]*u.f(xp0/p0*d1[2]+yb0/b0))+
               (pi[3]*u.f(xp0/p0*d1[3]+yb0/b0)))}

    U_xp0yb0.f <- function(xp0, yb0, K=par_xy.l$capital, p0=par_xy.l$price,
                           b0=par_xy.l$brice){
      mapply(function(xp, yb){
        if(K - xp - yb < 0){return(NA)}
        else{return(u.f(K - xp - yb) + bE_u1.f(xp, yb))}}, xp = xp0, yb = yb0)}

    par(mfrow = c(1, 2))



    U1.v <- U_xp0yb0.f(v.v, max_xy.l$maximum_y)

    if(max_xy.l$maximum_y > 0){
      U1.v[which.min(U1.v):length(U1.v)] <- NA}

    u1.v <- u.f(par_xy.l$capital - v.v - max_xy.l$maximum_y)
    u1.v[which.min(u1.v):length(u1.v)] <- NA

    bE_u1.v <- mapply(bE_u1.f, v.v, max_xy.l$maximum_y)


    fmax <- max(u.f(par_xy.l$capital - max_xy.l$maximum_x - max_xy.l$maximum_y),
                bE_u1.f(max_xy.l$maximum_x, max_xy.l$maximum_y),
                U_xp0yb0.f(max_xy.l$maximum_x, max_xy.l$maximum_y))
    fmin <- min(u.f(par_xy.l$capital - max_xy.l$maximum_x - max_xy.l$maximum_y),
                bE_u1.f(max_xy.l$maximum_x, max_xy.l$maximum_y),
                U_xp0yb0.f(max_xy.l$maximum_x, max_xy.l$maximum_y))

    xmax <- max(v.v)
    xmin <- min(v.v)
    ymax <- fmax + (fmax - fmin)
    ymin <- fmin - (fmax - fmin)
```

```
    plot(NA,NA,xaxs="r",yaxs="r",xlim=c(xmin,xmax), ylim=c(ymin, ymax),
         main=NA, xlab="xp0", ylab=NA, yaxt='n')

    legend("topright", c("U(xp0)", "u0", "beta*E[u1]"),
           fill = c("red", "blue", "green"))

    lines(v.v, U1.v, col="red")
    lines(v.v, u1.v,
          col = "blue")
    lines(v.v, bE_u1.v, col = "green")

    points(v.v[which.max(U1.v)], max(U1.v, na.rm = TRUE))




    U2.v <- U_xp0yb0.f(max_xy.l$maximum_x, v.v)

    if(max_xy.l$maximum_x > 0){
      U2.v[which.min(U2.v):length(U2.v)] <- NA}

    u2.v <- u.f(par_xy.l$capital - v.v - max_xy.l$maximum_x)
    u2.v[which.min(u2.v):length(u2.v)] <- NA

    bE_u1.v <- mapply(bE_u1.f, max_xy.l$maximum_x, v.v)

    plot(NA,NA,xaxs="r",yaxs="r",xlim=c(xmin,xmax), ylim=c(ymin, ymax),
         main=NA, xlab="yb0", ylab="U(c0,c1), u(c0), bE[u(c1)]", yaxt='n')

    legend("topright", c("U(yb0)", "u0", "beta*E[u1]"),
           fill = c("red", "blue", "green"))

    lines(v.v, U2.v, col="red")
    lines(v.v, u2.v,
          col = "blue")
    lines(v.v, bE_u1.v, col = "green")

    points(v.v[which.max(U2.v)], max(U2.v, na.rm = TRUE))



    title("PORTFOLIO CHOICE", line = -2, outer=TRUE)



  })



# isoquant plot riskless ---------------------------------------------------

  output$Isoquants_xy <- renderPlot({
    max_xy.l <- MaxExpTotUtility_xy()
    par_xy.l <- parameters_xy()
    pi.v <- probabilities_xy()
    d1.v <- dividends_xy()

    v.v <- par_xy.l$capital*x_axis_xy()
```

```
v.d <- c(min(v.v), max(v.v))

u.f <- function(c, gamma = par_xy.l$gamma){
  if(gamma == 1){return(log(c))}
  else{return((c^(1-gamma)-1)/(1-gamma))}}

bE_u1.f <- function(xp0, yb0, p0=par_xy.l$price, b0=par_xy.l$brice, pi=pi.v,
                    d1=d1.v, beta=par_xy.l$beta){
  return(beta*(pi[1]*u.f(xp0/p0*d1[1]+yb0/b0))+
           (pi[2]*u.f(xp0/p0*d1[2]+yb0/b0))+
           (pi[3]*u.f(xp0/p0*d1[3]+yb0/b0)))}

U_c0xp0.f <- function(xp0, yb0=max_xy.l$maximum_y, K=par_xy.l$capital,
                      p0=par_xy.l$price, b0=par_xy.l$brice){
  mapply(function(xp, yb){
    if(K - xp - yb < 0){return(NA)}
    else{return(u.f(K - xp - yb) + bE_u1.f(xp, yb))}},
      xp = xp0, yb = yb0)}

KK <- function(xp0, K=par_xy.l$capital, yb0=max_xy.l$maximum_y){
  return(K - xp0 - yb0)}

cons_f.f <- function(xp0, yb0=max_xy.l$maximum_y, K=par_xy.l$capital,
                     p0=par_xy.l$price, b0=par_xy.l$brice,
                     beta=par_xy.l$beta, gamma = par_xy.l$gamma,
                     f_const=max_xy.l$objective){
  if(gamma == 1){
    c0 <- exp(f_const - bE_u1.f(xp0, yb0))}
  else{c0 <- ((f_const - bE_u1.f(xp0, yb0))*(1-gamma)+1)^(1/(1-gamma))
    }
  return(c0)}


xmax <- 1.2*par_xy.l$capital
xmin <- 0
ymax <- 1.2*par_xy.l$capital
ymin <- 0

plot(NA, NA, xaxs="r", yaxs="r", xlim=c(xmin, xmax), ylim=c(ymin,ymax),
     main="ISOQUANTS", xlab="xp0", ylab="c0")

vK.v <- head(v.v, which.min(abs(KK(v.v))))
lines(vK.v, KK(vK.v))

v.v2 <- v.v
cons_f.v <- cons_f.f(v.v2)
cons_f.v[1: which.max(cons_f.v)] <- NA
cons_f.v[cons_f.v > par_xy.l$capital] <- NA

k <- which.min(abs(cons_f.v - par_xy.l$capital))
cons_f.v[k - 1] <- par_xy.l$capital
v.v2[k-1] <- v.v2[k] + (v.v2[k] - v.v2[k+1])/(cons_f.v[k]-cons_f.v[k+1])*
  (cons_f.v[k-1]-cons_f.v[k])

lines(v.v2, cons_f.v)
```

```
    delta_yb0 <- max_xy.l$maximum_y * par_xy.l$delta
    yb0_new <- max_xy.l$maximum_y * (1 + par_xy.l$delta)

    KK.v <- KK(v.v, K= par_xy.l$capital, yb0 = yb0_new)
    KK.v[KK.v < 0] <- NA
    lines(v.v, KK.v, lty = "dashed")

    U_c0xp0.v <- U_c0xp0.f(v.v, yb0 = yb0_new)
    f_new <- max(U_c0xp0.v, na.rm = TRUE)
    xp0_new <- which.max(U_c0xp0.v)

    v.v3 <- v.v
    cons_f2.v <- cons_f.f(v.v, yb0=yb0_new, f_const=f_new)

    cons_f2.v[1: which.max(cons_f2.v)] <- NA
    cons_f2.v[cons_f2.v > par_xy.l$capital] <- NA

    n <- which.min(abs(cons_f2.v - par_xy.l$capital))
    cons_f2.v[n - 1] <- par_xy.l$capital
    v.v3[n-1] <- v.v3[n] + (v.v3[n] - v.v3[n+1])/(cons_f2.v[n]-cons_f2.v[n+1])*
      (cons_f2.v[n-1]-cons_f2.v[n])

    lines(v.v3, cons_f2.v, lty = "dashed")

    points(max_xy.l$maximum_x,
           par_xy.l$capital - max_xy.l$maximum_x - max_xy.l$maximum_y)

    legend("topright", c(c("x*p0 =", "c0* =", "", "y*b0 =", "yb0_new =", "",
                           "U* =", "U*_new ="),
          c(round(max_xy.l$maximum_x, log10(max_xy.l$maximum_x)+3),
          round(par_xy.l$capital - max_xy.l$maximum_x -max_xy.l$maximum_y,
          log10(par_xy.l$capital - max_xy.l$maximum_x -max_xy.l$maximum_y)+3),
          "", round(max_xy.l$maximum_y, log10(max_xy.l$maximum_y)+3),
          round(yb0_new, log10(abs(yb0_new))+3), "",
          round(max_xy.l$objective, log10(max_xy.l$objective)+3),
          round(f_new, log10(f_new)+3))),
          ncol = 2)
  })


# marginal utility plot riskless ------------------------------------------------

  output$MarginalUtility_xy <- renderPlot({
    max_xy.l <- MaxExpTotUtility_xy()
    par_xy.l <- parameters_xy()
    pi.v <- probabilities_xy()
    d1.v <- dividends_xy()

    v.v <- par_xy.l$capital*x_axis_xy()
    v.d <- c(min(v.v), max(v.v))

    u.f <- function(c, gamma = par_xy.l$gamma){
      if(gamma == 1){return(log(c))}
      else{return((c^(1-gamma)-1)/(1-gamma))}}

    bE_u1.f <- function(xp0, yb0, p0=par_xy.l$price, b0=par_xy.l$brice, pi=pi.v,
                        d1=d1.v, beta=par_xy.l$beta){
      return(beta*(pi[1]*u.f(xp0/p0*d1[1]+yb0/b0))+
               (pi[2]*u.f(xp0/p0*d1[2]+yb0/b0))+
```

```
            (pi[3]*u.f(xp0/p0*d1[3]+yb0/b0)))}

U_c0.f <- function(c0, xp0=max_xy.l$maximum_x, yb0=max_xy.l$maximum_y,
                   K=par_xy.l$capital, p0=par_xy.l$price, b0=par_xy.l$brice)
  {return(u.f(c0) + bE_u1.f(xp0, yb0))}

U_xp0.f <- function(xp0, yb0=max_xy.l$maximum_y, K=par_xy.l$capital,
                    p0=par_xy.l$price, b0=par_xy.l$brice)
  {return(u.f(K - max_xy.l$maximum_x - yb0) + bE_u1.f(xp0, yb0))}

U_yb0.f <- function(yb0, xp0=max_xy.l$maximum_x, K=par_xy.l$capital,
                    p0=par_xy.l$price, b0=par_xy.l$brice)
  {return(u.f(K - max_xy.l$maximum_y - xp0) + bE_u1.f(xp0, yb0))}

xmax <- par_xy.l$capital
xmin <- 0

# fit y_axis limits to the plot
p <- par_xy.l$capital - max_xy.l$maximum_x - max_xy.l$maximum_y

p1 <- max(xmin, p - (xmax-xmin)/3, na.rm = TRUE)
p2 <- min(xmax, p + (xmax-xmin)/3, na.rm = TRUE)
slope <- (U_c0.f(p+0.01*xmax)-U_c0.f(p-0.01*xmax))/(0.02*xmax)
yaxismax <- U_c0.f(p) + (p2-p)*slope
yaxismin <- U_c0.f(p) + (p1-p)*slope
yaxisdist <- yaxismax - yaxismin
ymax <- yaxismax + yaxisdist
ymin <- yaxismin - yaxisdist

par(mfrow = c(1, 3))
plot(NA,NA,xaxs="r",yaxs="r",xlim=c(xmin, xmax), ylim=c(ymin, ymax),
     main="marginal utility c0", xlab="c0", ylab="", yaxt='n')

lines(v.v, U_c0.f(v.v))

# tangent to immediate consumption part
tangent_U_c0.f <- function(p){
  x1 <- max(xmin, p - (xmax-xmin)/3, na.rm = TRUE)
  x2 <- min(xmax, p + (xmax-xmin)/3, na.rm = TRUE)
  slope <- (U_c0.f(p+0.01*xmax)-U_c0.f(p-0.01*xmax))/(0.02*xmax)
  U_c0.f1 <- U_c0.f(p) + (x1-p)*slope
  U_c0.f2 <- U_c0.f(p) + (x2-p)*slope
  lines(c(x1, x2), c(U_c0.f1, U_c0.f2))
  points(p, U_c0.f(p))
  return(slope)}

s <- tangent_U_c0.f(par_xy.l$capital-max_xy.l$maximum_x-max_xy.l$maximum_y)
s <- round(s, abs(log10(s))+1)

legend("bottomright", c("dU/dc0 = ", s))



plot(NA,NA,xaxs="r",yaxs="r",xlim=c(xmin, xmax), ylim=c(ymin, ymax),
     main="marginal utility beta*E[u(xp0)]", xlab="xp0", ylab="", yaxt='n')

lines(v.v, U_xp0.f(v.v))

# tangent to risky investment part
tangent_U_xp0.f <- function(p){
```

```
      x1 <- max(xmin, p - (xmax-xmin)/3, na.rm = TRUE)
      x2 <- min(xmax, p + (xmax-xmin)/3, na.rm = TRUE)
      slope <- (U_xp0.f(p+0.01*xmax)-U_xp0.f(p-0.01*xmax))/(0.02*xmax)
      U_xp0.f1 <- U_xp0.f(p) + (x1-p)*slope
      U_xp0.f2 <- U_xp0.f(p) + (x2-p)*slope
      lines(c(x1, x2), c(U_xp0.f1, U_xp0.f2))
      points(p, U_xp0.f(p))
      return(slope)}

    t <- tangent_U_xp0.f(max_xy.l$maximum_x)
    t <- round(t, abs(log10(t))+1)

    legend("bottomright", c("dU/dxp0 = ", t))


    plot(NA,NA,xaxs="r",yaxs="r",xlim=c(xmin, xmax), ylim=c(ymin, ymax),
         main="marginal utility beta*E[u(yb0)]", xlab="yb0", ylab="", yaxt='n')

    lines(v.v, U_yb0.f(v.v))

    # tangent to riskless investment part
    tangent_U_yb0.f <- function(p){
      x1 <- max(xmin, p - (xmax-xmin)/3, na.rm = TRUE)
      x2 <- min(xmax, p + (xmax-xmin)/3, na.rm = TRUE)
      slope <- (U_yb0.f(p+0.01*xmax)-U_yb0.f(p-0.01*xmax))/(0.02*xmax)
      U_yb0.f1 <- U_yb0.f(p) + (x1-p)*slope
      U_yb0.f2 <- U_yb0.f(p) + (x2-p)*slope
      lines(c(x1, x2), c(U_yb0.f1, U_yb0.f2))
      points(p, U_yb0.f(p))
      return(slope)}

    u <- tangent_U_yb0.f(max_xy.l$maximum_y)
    u <- round(u, abs(log10(u))+1)

    legend("bottomright", c("dU/dyb0 = ", u))
  })



# results output riskless ---------------------------------------------------

  output$maxZf_xy <- renderText({
    max_xy.l <- MaxExpTotUtility_xy()
    paste("max. exp. Utility: ", round(max_xy.l$objective, 3))
  })

  output$max_xy_x <- renderText({
    max_xy.l <- MaxExpTotUtility_xy()
    paste("maximum in x*p0: ", round(max_xy.l$maximum_x, 3))
  })

  output$max_xy_y <- renderText({
    max_xy.l <- MaxExpTotUtility_xy()
    paste("maximum in y*b0: ", round(max_xy.l$maximum_y, 3))
  })
}
```