Example of thermod application: Lake Mendota

Robert Ladwig

6/18/2021

thermod-Package

thermod is a simple two-layer heat transport model that assumes that the lake is divided into two completely mixed volumes: the epilimnion and the hypolimnion. Both layers are divided by a thermocline zone. The entrainment over the thermocline depends on a diffusion coefficient which is a function of the diffusion at neutral stability to the Richardson number. All equations and derivations are based on or from Steven C. Chapra (2008) 'Surface Water-Quality Modeling' Waveland Press, Inc.

thermod allows you to easily configure a lake setup from a LakeEnsemblR-configuration file:

```
# use a LakeEnsemblR setup
config_file <- 'LakeEnsemblR.yaml'
folder = '.'
parameters <- configure_from_ler(config_file <- config_file, folder = folder)

# load in the meteorological boundary data
bound <-read_delim(paste0(folder,'/meteo.txt'), delim = '\t')
bound <- bound[, -c(1)]
colnames(bound) <- c('Day','Jsw','Tair','Dew','vW')

# function to calculate wind shear stress (and transforming wind speed from km/h to m/s)
bound$Uw <- 19.0 + 0.95 * (bound$vW * 1000/3600)^2
bound$vW <- bound$vW * 1000/3600

boundary <- bound
```

thermod then solves its ordnary differential equations on a vertical grid with two elements and on a daily time step using the 4th-order Runge-Kutta method: $c_{i+1} = c_i + [\frac{1}{6}(k_1 + 2k_2 + k_3 + k_4)]h$. Further, the depth of the thermocline is fixed and either provided by the user or estimated based on maximum length or width of the basin using the empirical eqation from Hanna M. (1990) Evaluation of Models predicting Mixing depth. Can. J. Fish. Aquat. Sci. 47: 940-947: $z_{therm} = 10^{0.336log10(max(L,W))-0.245}$.

Temperature simulation

thermod has three main modes: (1) temperature, (2) temperature and dissolved oxygen, and (3) temperature, dissolved oxygen and nutrient-food web model (NPZ). Let us first simulate only water temperatures in both layers for Lake Mendota, and compare this with observed data.

Here, thermod solves two ordinary differential equations for water temperature: $\frac{dT_{epi}}{dt} = \frac{Q}{V_{epi}}T_{in} - \frac{Q}{V_{epi}}T_{epi} + \upsilon_t \frac{A_t}{V_{epi}}(T_{hypo} - T_{epi}) \pm \frac{A_{surf}}{V_{epi}\rho c_p}J$ and $\frac{dT_{hypo}}{dt} = \upsilon_t \frac{A_t}{V_{hypo}}(T_{epi} - T_{hypo})$

The atmospheric heat fluxes, J, are separated into the five sub-heat fluxes shortwave radiation, incominglong-wave radiation, outgoing longwace radiation, latent heat flux and sensible heat flux: $J = J_{sw} + J_{in} * J_{out} + J_E + J_H$. Here, J_{sw} is a measured bondary condition, whereas the other heat terms are estimated from meteorolo-

gical data: $J_{in} = \sigma(T_{air} + 273)^4(K + 0.031\sqrt{e_{air}})(1 - R_L)$, $J_{out} = \epsilon\sigma(T_{epi} + 273)^4$, $J_E = f(Uw)(e_{sat} - e_{air})$ with $f(Uw) = 19.0 + 0.95{u_{wind}}^2$, and $J_H = c_1 f(Uw)(T_{epi} - T_{air})$.

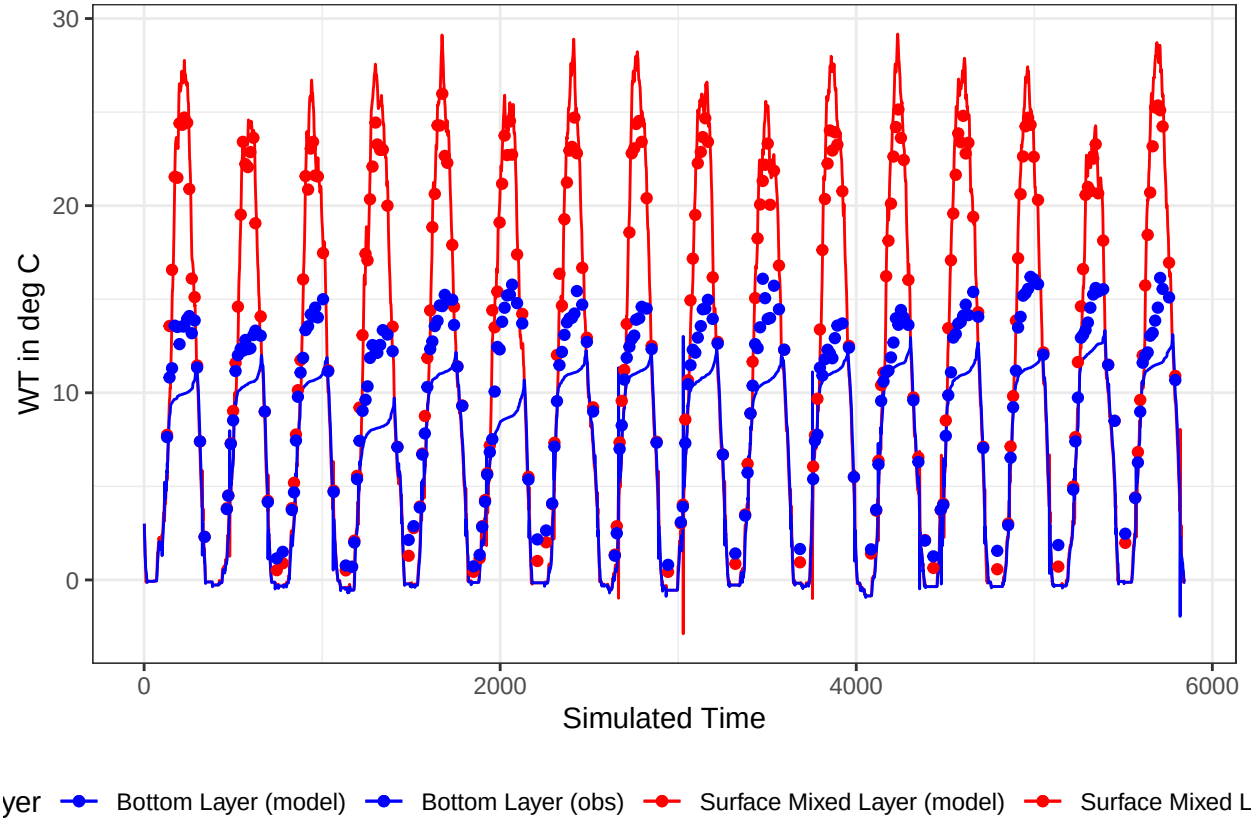
thermod aims to get a dynamic representation of the vertical diffusivity in dependence of the water column stability (in contrast to telling the model to use high or low values on specific days). Therefore, the model states the diffusion coefficient as being dependent on the diffusion at neutral stability to the Richardson number: $\upsilon_t = \frac{E_0}{(1+\alpha R_i)^{3/2}}$. The diffusion coefficient at neutral stability then depends on the ratio of wind shear stress to water density: $E_0 = c\sqrt{\frac{\tau}{\rho_{water}}}$ with wind shear stress as: $\tau = \rho_{air} C_d {u_{wind}}^2$. Finally, we can estimate the Richardson number, which generally describes the work of buoyancy against wind-induced turbulence. If it is below 1/4, the system will experience shear-induced turbulence: $Ri = \frac{-\frac{g}{\rho}\frac{d\rho}{dz}}{\sqrt{\frac{\tau}{\rho_{water}}}H^{-2}}$ which acts over a thickness H which we assume to be the thermocline depth.

```
# simulation maximum length
times <- seq(from = 1, to = max(boundary$Day), by = 1)

# initial water temperatures
yini <- c(3,3)

# there is calibration parameter for the vertical diffusivity over the thermocline, which we
# will set slightly higher to 3.2
parameters[19] = 3.2

# you can also set ice_on to TRUE to reduce atmospheric exchanges during idealized ice conditions
ice_on = TRUE

out <- run_model(bc = boundary, params = parameters, ini = yini, times = times, ice = ice_on)

result <- data.frame('Time' = out[,1],
                     'WT_epi' = out[,2], 'WT_hyp' = out[,3])
head(result)
```

```
##   Time   WT_epi   WT_hyp
## 1    1 3.000000 3.000000
## 2    2 2.903372 2.412467
## 3    3 2.499375 2.234035
## 4    4 2.174156 1.831215
## 5    5 1.749387 1.488693
## 6    6 1.344808 1.221217
```

```
# load observed data
obs <-read_delim(paste0('obs.txt'), delim = ',')
simple_therm_depth = parameters[18]
start_date <- get_yaml_value(config_file, "time", "start")
stop_date <- get_yaml_value(config_file, "time", "stop")

obs_sfc <- obs %>%
  filter(Depth_meter <= simple_therm_depth) %>%
  mutate('date' = yday(datetime),
         'sfc' = Water_Temperature_celsius) %>%
  group_by(datetime) %>%
  summarise('wtr_avg' = mean(sfc, na.rm = TRUE))
obs_sfc$time = match(as.Date(obs_sfc$datetime), seq(as.Date(start_date), as.Date(stop_date), by = 'day'
obs_btm <- obs %>%
  filter(Depth_meter >= simple_therm_depth) %>%
```

```
  mutate('date' = yday(datetime),
         'btm' = Water_Temperature_celsius) %>%
  group_by(datetime) %>%
  summarise('wtr_avg' = mean(btm, na.rm = TRUE))
obs_btm$time = match(as.Date(obs_btm$datetime), seq(as.Date(start_date), as.Date(stop_date), by = 'day'

ggplot(result) +
  geom_line(aes(x=Time, y=WT_epi, col='Surface Mixed Layer (model)')) +
  geom_line(aes(x=(Time), y=WT_hyp, col='Bottom Layer (model)')) +
  geom_point(data = obs_sfc, aes(x=time, y=wtr_avg, col='Surface Mixed Layer (obs)'),linetype = "dashed
  geom_point(data = obs_btm, aes(x=(time), y=wtr_avg, col='Bottom Layer (obs)'), linetype = "dashed") +
  labs(x = 'Simulated Time', y = 'WT in deg C')  +
  scale_color_manual(values = c('blue','blue','red','red')) +
  theme_bw()+
  guides(col=guide_legend(title="Layer")) +
  theme(legend.position="bottom")
```



Dissolved oxygen simulation

We can now also add dissolved oxygen equations to the mix, where net ecosystem production and sediment oxygen demand are temperature-dependet but also fitting parameters (here using P_{NEP} and P_{SED}): $\frac{dDO_{epi}}{dt} = F_{NEP} + F_{ATM} + \upsilon_{t,DO} * A_T * (\frac{DO_{hypo}}{V_{hypo}} - \frac{DO_{epi}}{V_{epi}})$ with $F_{ATM} = k(DO_{sat} - \frac{DO_{epi}}{V_{epi}}) * A_{surf}$ (gas exchange after the method according to Cole (1998)), and $N_{NEP} = \theta^{T_{epi}-20} * P_{NEP} * V_{epi}$; and $\frac{dDO_{hypo}}{dt} = \upsilon_{t,DO} * A_T * (\frac{DO_{epi}}{V_{epi}} - \frac{DO_{hypo}}{V_{hypo}})$ with $F_{SED} = P_{SED} * A_{SED} * \theta^{T_{hypo}-20}(\frac{\frac{DO_{hypo}}{V_{hypo}}}{k_{1/2}+\frac{DO_{hypo}}{V_{hypo}}})$. The parameter $\upsilon_{t,DO}$ depends on the estimated vertical diffusivity but is set several magnitudes lower. Note that thermod itself tracks the

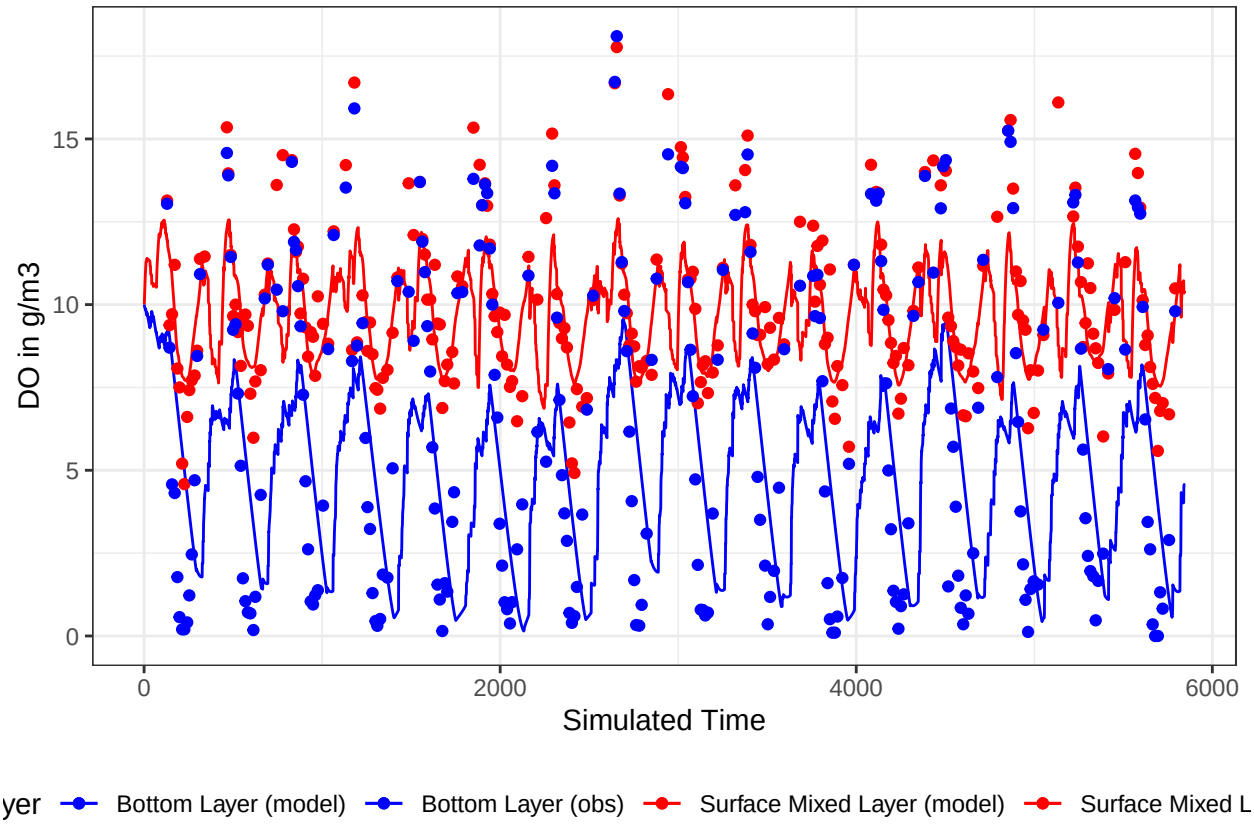
mass of dissolved oxygen per layer.

```
# we need to configure additional parameters for oxygen dynamics first
wq_parameters <- append(parameters, c(0.001 / 1000,
                                      100, 15000 * 1e4, 100))

wq_parameters[19] = parameters[19]

# simulation maximum length
times <- seq(from = 1, to = max(boundary$Day), by = 1)

# initial water temperatures and oxygen conditions
yini <- c(3,3, 10 * 1000/1e6  * wq_parameters[1], 10 * 1000/1e6  * wq_parameters[2])

ice_on = TRUE
out <- run_oxygen_model(bc = boundary, params = wq_parameters, ini = yini, times = times, ice = ice_on)

result <- data.frame('Time' = out[,1],
                     'WT_epi' = out[,2], 'WT_hyp' = out[,3],
                     'DO_epi' = out[,4], 'DO_hyp' = out[,5])
head(result)
```

```
##   Time   WT_epi   WT_hyp       DO_epi       DO_hyp
## 1    1 3.000000 3.000000 2.743250e+12 2.125000e+12
## 2    2 2.903372 2.412467 2.789142e+12 2.116976e+12
## 3    3 2.499375 2.234035 2.832751e+12 2.109784e+12
## 4    4 2.174156 1.831215 2.872690e+12 2.106131e+12
## 5    5 1.749387 1.488693 2.912628e+12 2.102348e+12
## 6    6 1.344808 1.221217 2.952660e+12 2.097875e+12
```

```
# load observed oxygen data
obs <-read_delim(paste0('obs_oxygen.txt'), delim = ',')
simple_therm_depth = parameters[18]
start_date <- get_yaml_value(config_file, "time", "start")
stop_date <- get_yaml_value(config_file, "time", "stop")

obs_sfc <- obs %>%
  filter(Depth_meter <= simple_therm_depth) %>%
  mutate('date' = yday(datetime),
         'sfc' = Dissolved_Oxygen_gPerCubicMeter ) %>%
  group_by(datetime) %>%
  summarise('do_avg' = mean(sfc, na.rm = TRUE))
obs_sfc$time = match(as.Date(obs_sfc$datetime), seq(as.Date(start_date), as.Date(stop_date), by = 'day'))
obs_btm <- obs %>%
  filter(Depth_meter >= simple_therm_depth) %>%
  mutate('date' = yday(datetime),
         'btm' = Dissolved_Oxygen_gPerCubicMeter ) %>%
  group_by(datetime) %>%
  summarise('do_avg' = mean(btm, na.rm = TRUE))
obs_btm$time = match(as.Date(obs_btm$datetime), seq(as.Date(start_date), as.Date(stop_date), by = 'day'))

ggplot(result) +
  geom_line(aes(x=Time, y=DO_epi / 1000 /  wq_parameters[1] * 1e6, col='Surface Mixed Layer (model)')) +
  geom_line(aes(x=(Time), y=DO_hyp / 1000 /  wq_parameters[2] * 1e6, col='Bottom Layer (model)')) +
  geom_point(data = obs_sfc, aes(x=time, y=do_avg, col='Surface Mixed Layer (obs)'),linetype = "dashed")
```

```
  geom_point(data = obs_btm, aes(x=(time), y=do_avg, col='Bottom Layer (obs)'),linetype = "dashed") +
  labs(x = 'Simulated Time', y = 'DO in g/m3')  +
  scale_color_manual(values = c('blue','blue','red','red')) +
  theme_bw()+
  guides(col=guide_legend(title="Layer")) +
  theme(legend.position="bottom")
```



NPZ (nutrient-phytoplankton-zooplankton) simulation

We can also simulate a simplified NPZ model using thermod, where we solve 5 ordinary differential equations per layer (so, 10 in total). All these ODEs are coupled to each other, e.g. - all equations are temperature-dependent - phytoplankton are grazed by zooplankton - zooplankton die and get converted to nutrients (i.e., phosphorus) - nutrients get converted into phytoplankton, and vice versa - net ecosystem production of oxygen depends on phytoplankton - sediment flux of nutrients depend on dissolved oxygen

All equations depend on a-parameters that transform mass from one state to the other, e.g. chlorophyll-a, carbon or phosphorus. Further, we state that phytoplankton and zooplankton are sinking down towards the sediment, whereas phosphorus can also be transported between the two layers via turbulent diffusion. First, we state the ODEs for phytoplankton as the mass of chlorophyll-a: $\frac{dPHY_{epi}}{dt} = [(k_g \frac{P_{epi}}{k_{1/2}+P_{epi}} - k_{ra}) * PHY_{epi} - C_{gz} * ZOO_{epi} * PHY_{epi}]\theta^{T_{epi}-20} - k_{sa} * PHY_{epi}$ and $\frac{dPHY_{hypo}}{dt} = [(k_g \frac{P_{hypo}}{k_{1/2}+P_{hypo}} - k_{ra}) * PHY_{hypo} - C_{gz} * ZOO_{hypo} * PHY_{hypo}]\theta^{T_{hypo}-20} + k_{sa} * PHY_{epi} - k_{sa} * PHY_{hypo}$. Here, phytoplankton depends on a growth and respiration difference, grazing by zooplankton, and a sinking term. Then we can write the changes of zooplankton as changes of carbon (C) mass: $\frac{dZOO_{epi}}{dt} = [(a_{ca} * \epsilon * C_{gz}) * ZOO_{epi} * PHY_{epi}) - k_{rz} * PHY_{epi}]\theta^{T_{epi}-20} - k_{sz} * ZOO_{epi}$ and $\frac{dZOO_{hypo}}{dt} = [(a_{ca} * \epsilon * C_{gz}) * ZOO_{hypo} * PHY_{hypo}) - k_{rz} * PHY_{hypo}]\theta^{T_{hypo}-20} + k_{sz} * ZOO_{epi} - k_{sz} * ZOO_{hypo}$. Here, zooplankton depends on grazing of phytoplankton, a respiration to phosphorus, and a sinking term. Finaly, the mass of phosphorus changes using: $\frac{dP_{epi}}{dt} = [a_{pa}(1-\epsilon) * C_{gz} * ZOO_{epi} *$

$PHY_{epi} + a_{pc} * k_{rz} * ZOO_{epi} - a_{pa}(k_g \frac{P_{epi}}{k_{1/2}+P_{epi}} - k_{ra})PHY_{epi}]\theta^{T_{epi}-20} + \upsilon_{t,DO} * A_t * (\frac{P_{hypo}}{V_{hypo}} - \frac{P_{epi}}{V_{epi}})$ and $\frac{dP_{hypo}}{dt} = [a_{pa}(1-\epsilon)*C_{gz}*ZOO_{hypo}*PHY_{hypo}+a_{pc}*k_{rz}*ZOO_{epi}-a_{pa}(k_g \frac{P_{hypo}}{k_{1/2}+P_{hypo}}-k_{ra})PHY_{hypo}]\theta^{T_{hypo}-20}+ \upsilon_{t,DO} * A_t * (\frac{P_{epi}}{V_{epi}} - \frac{P_{hypo}}{V_{hypo}}) - P_{SED,P} * A_{SED}\theta^{T_{hypo}-20} * (\frac{\frac{DO_{hypo}}{V_{hypo}}}{k_{1/2}+\frac{DO_{hypo}}{V_{hypo}}})$. Here, phosphorus is gained from grazing of zooplankton on phytoplankton, from the respiration of zooplankton as well as phytoplankton, from turbulent diffusion, and (in the hypolimnion) from a sediment flux. Finally, we can couple the dynamics of phytoplankton to dissolved oxygen now by removing the fitting parameter P_{NEP} and quantifying it as: $F_{NEP} = \theta^{T_{epi}-20}(\alpha_1 * k_g \frac{P_{epi}}{k_{1/2}+P_{epi}} - \alpha_2 * k_{ra})PHY_{epi}$.

These additional layers of complexity result in a vast increase of additional fitting parameters, as well as amore profound effect of the initial conditions on the model run.

```
npz_specific <- c(0.165, 0.15, 1.5*1e3, 1e-5, 0.04 * 1000, 0.6, 0.1 , 1e-5, 15, 0.3, -1500 / 1e4, 1e-15

npz_parameters <- append(parameters, c(0.001 / 1000,
                                       100, 15000 * 1e4, 100,
                                       npz_specific))

npz_parameters[19] = parameters[19]

# simulation maximum length
times <- seq(from = 1, to = max(boundary$Day), by = 1)

# initial water temperatures, oxygen, phytoplankton, zooplankton and nutrient conditions
yini <- c(3,3, 10 * 1000/1e6  * wq_parameters[1], 10 * 1000/1e6  * wq_parameters[2],
          1/1e6  * wq_parameters[1], 1/1e6  * wq_parameters[2],
          0.05 * 1000/1e6  * wq_parameters[1], 0.05 * 1000/1e6  * wq_parameters[2],
          20 /1000/(0.001 * 10e6) * wq_parameters[1], 20 /1000/(0.001 * 10e6) * wq_parameters[2])

ice_on = TRUE

out <- run_npz_model(bc = boundary, params = npz_parameters, ini = yini, times = times, ice = ice_on)

result <- data.frame('Time' = out[,1],
                     'WT_epi' = out[,2], 'WT_hyp' = out[,3],
                     'DO_epi' = out[,4] / 1000 /  wq_parameters[1] * 1e6,
                     'DO_hyp' = out[,5] / 1000 /  wq_parameters[1] * 1e6,
                     'PHY_epi' = out[,6] / 1000 /  wq_parameters[1] * 1e6,
                     'PHY_hyp' = out[,7] / 1000 /  wq_parameters[1] * 1e6,
                     'ZOO_epi' = out[,8] / 1000 /  wq_parameters[1] * 1e6,
                     'ZOO_hyp' = out[,9] / 1000 /  wq_parameters[1] * 1e6,
                     'P_epi' = out[,10] / 1000 /  wq_parameters[1] * 1e6,
                     'P_hyp' = out[,11] / 1000 /  wq_parameters[1] * 1e6)
head(result)
```

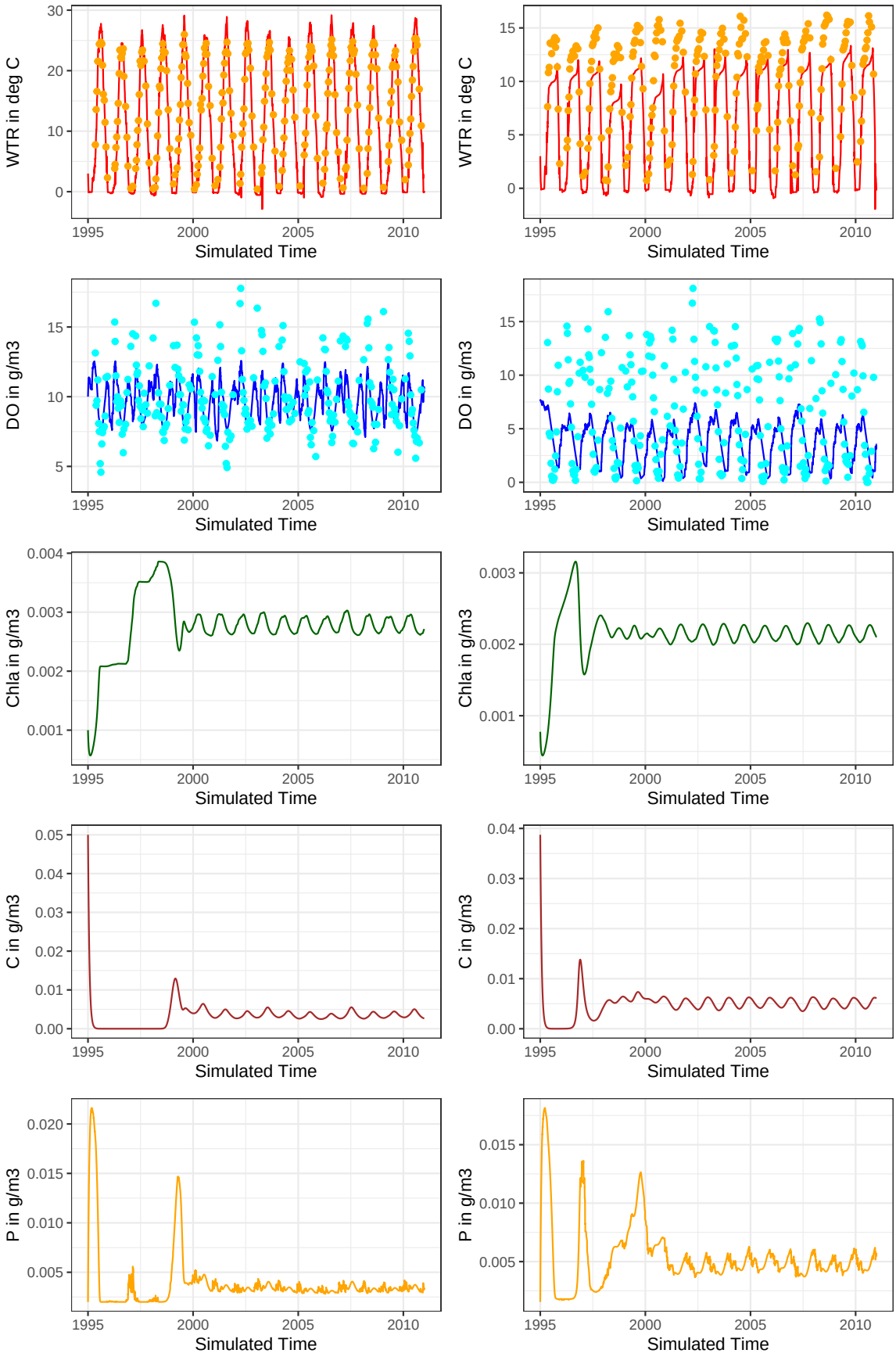
```
##   Time   WT_epi   WT_hyp   DO_epi   DO_hyp      PHY_epi      PHY_hyp    ZOO_epi
## 1    1 3.000000 3.000000 10.00000 7.746286 0.0010000000 0.0007746286 0.05000000
## 2    2 2.903372 2.412467 10.16671 7.717031 0.0009582751 0.0007424691 0.04808396
## 3    3 2.499375 2.234035 10.32514 7.690799 0.0009220934 0.0007146250 0.04621851
## 4    4 2.174156 1.831215 10.47027 7.677437 0.0008899802 0.0006899416 0.04440879
## 5    5 1.749387 1.488693 10.61541 7.663602 0.0008612362 0.0006678300 0.04266201
## 6    6 1.344808 1.221217 10.76092 7.647263 0.0008353531 0.0006478634 0.04098025
##      ZOO_hyp       P_epi       P_hyp
## 1 0.03873143 0.002000000 0.001549257
```

```
## 2 0.03725257 0.003122780 0.002461422
## 3 0.03581381 0.004154579 0.003300796
## 4 0.03442002 0.005117824 0.004080047
## 5 0.03307416 0.006016819 0.004810173
## 6 0.03177512 0.006857549 0.005497960
```

```
time_seq <- seq.Date(from =   as.Date(get_yaml_value(config_file, "time", "start")),
                     to = as.Date(get_yaml_value(config_file, "time", "stop")),
                     by = 'day')
result$Date = time_seq

obs_sfc <- obs %>%
  filter(Depth_meter <= simple_therm_depth) %>%
  mutate('date' = yday(datetime),
         'sfc' = Water_Temperature_celsius) %>%
  group_by(datetime) %>%
  summarise('wtr_avg' = mean(sfc, na.rm = TRUE))
obs_sfc$time = match(as.Date(obs_sfc$datetime), seq(as.Date(start_date), as.Date(stop_date), by = 'day'
obs_btm <- obs %>%
  filter(Depth_meter >= simple_therm_depth) %>%
  mutate('date' = yday(datetime),
         'btm' = Water_Temperature_celsius) %>%
  group_by(datetime) %>%
  summarise('wtr_avg' = mean(btm, na.rm = TRUE))
obs_btm$time = match(as.Date(obs_btm$datetime), seq(as.Date(start_date), as.Date(stop_date), by = 'day'

g1 <- ggplot(result) +
  geom_line(aes(Date, WT_epi), col = 'red') +
  labs(x = 'Simulated Time', y = 'WTR in deg C')  +
  scale_x_date(limits= as.Date(c(min(result$Date), max(result$Date))))+
  geom_point(data = obs_sfc, aes(x=as.Date(datetime), y=wtr_avg), col = 'orange') +
  theme_bw()+
  theme(legend.position="bottom")
g2 <- ggplot(result) +
  geom_line(aes(Date, WT_hyp), col = 'red') +
  labs(x = 'Simulated Time', y = 'WTR in deg C')  +
  theme_bw()+
  geom_point(data = obs_btm, aes(x=as.Date(datetime), y=wtr_avg), col = 'orange') +
  scale_x_date(limits= as.Date(c(min(result$Date), max(result$Date))))+
  theme(legend.position="bottom")

obs_sfc <- obs %>%
  filter(Depth_meter <= simple_therm_depth) %>%
  mutate('date' = yday(datetime),
         'sfc' = Dissolved_Oxygen_gPerCubicMeter ) %>%
  group_by(datetime) %>%
  summarise('do_avg' = mean(sfc, na.rm = TRUE))
obs_sfc$time = match(as.Date(obs_sfc$datetime), seq(as.Date(start_date), as.Date(stop_date), by = 'day'
obs_btm <- obs %>%
  filter(Depth_meter >= simple_therm_depth) %>%
  mutate('date' = yday(datetime),
         'btm' = Dissolved_Oxygen_gPerCubicMeter ) %>%
  group_by(datetime) %>%
  summarise('do_avg' = mean(btm, na.rm = TRUE))
```

```r
obs_btm$time = match(as.Date(obs_btm$datetime), seq(as.Date(start_date), as.Date(stop_date), by = 'day'))

g3 <- ggplot(result) +
  geom_line(aes(Date, DO_epi), col = 'blue') +
  labs(x = 'Simulated Time', y = 'DO in g/m3')  +
  geom_point(data = obs_sfc, aes(x=as.Date(datetime), y=do_avg), col = 'cyan') + # sfc
  scale_x_date(limits= as.Date(c(min(result$Date), max(result$Date))))+
  theme_bw()+
  theme(legend.position="bottom")
g4 <- ggplot(result) +
  geom_line(aes(Date, DO_hyp), col = 'blue') +
  geom_point(data = obs_btm, aes(x=as.Date(datetime), y=do_avg), col = 'cyan') + # btm
  scale_x_date(limits= as.Date(c(min(result$Date), max(result$Date))))+
  labs(x = 'Simulated Time', y = 'DO in g/m3')  +
  theme_bw()+
  theme(legend.position="bottom")

g5 <- ggplot(result) +
  geom_line(aes(Date, PHY_epi), col = 'darkgreen') +
  labs(x = 'Simulated Time', y = 'Chla in g/m3')  +
  scale_x_date(limits= as.Date(c(min(result$Date), max(result$Date))))+
  theme_bw()+
  theme(legend.position="bottom")
g6 <- ggplot(result) +
  geom_line(aes(Date, PHY_hyp), col = 'darkgreen') +
  labs(x = 'Simulated Time', y = 'Chla in g/m3')  +
  scale_x_date(limits= as.Date(c(min(result$Date), max(result$Date))))+
  theme_bw()+
  theme(legend.position="bottom")

g7 <- ggplot(result) +
  geom_line(aes(Date, ZOO_epi), col = 'brown') +
  labs(x = 'Simulated Time', y = 'C in g/m3')  +
  scale_x_date(limits= as.Date(c(min(result$Date), max(result$Date))))+
  theme_bw()+
  theme(legend.position="bottom")
g8 <- ggplot(result) +
  geom_line(aes(Date, ZOO_hyp), col = 'brown') +
  labs(x = 'Simulated Time', y = 'C in g/m3')  +
  scale_x_date(limits= as.Date(c(min(result$Date), max(result$Date))))+
  theme_bw()+
  theme(legend.position="bottom")

g9 <- ggplot(result) +
  geom_line(aes(Date, P_epi), col = 'orange') +
  labs(x = 'Simulated Time', y = 'P in g/m3')  +
  scale_x_date(limits= as.Date(c(min(result$Date), max(result$Date))))+
  theme_bw()+
  theme(legend.position="bottom")
g10 <- ggplot(result) +
  geom_line(aes(Date, P_hyp), col = 'orange') +
  labs(x = 'Simulated Time', y = 'P in g/m3')  +
  scale_x_date(limits= as.Date(c(min(result$Date), max(result$Date))))+
```

```
  theme_bw()+
  theme(legend.position="bottom")

(g1 / g3 / g5 / g7 / g9) | (g2 / g4 / g6 / g8 / g10)  +
  plot_annotation(
    title = 'Thermod, 2-layer NPZ model, v1.01',
    subtitle = 'Epilimnion (left), Hypolimnion (right)',
    caption = 'Under development'
  )
```


WTR in deg C
DO in g/m3
Chla in g/m3
C in g/m3
P in g/m3
Simulated Time
1995
2000
2005
2010

We can inspect seasonal dynamics more closely by scaling the modeled variables. Surface water temperature peak in summer, whereas dissolved oxygen goes down over the course of the summer season. Oxygen seems to peak twice: once during fall mixing when oxygen gets replenished through the water column, and once during the peak of phytoplankton in spring. Meanwhile, nutrients (phosphate) peak first (then decline), followed by a peak of phytoplankton and, afterwards, a peak of zooplankton.

```
ggplot(result) +
  geom_line(aes(Date, scale(WT_epi), col = 'WTR'), col = 'red') +
  labs(x = 'Simulated Time', y = 'Scaled')  +
  geom_line(aes(Date, scale(DO_epi), col = 'DO'), col ='blue') +
  geom_line(aes(Date, scale(PHY_epi), col = 'PHY'), col = 'darkgreen') +
  geom_line(aes(Date, scale(ZOO_epi), col = 'ZOO'), col = 'brown') +
  geom_line(aes(Date, scale(P_epi), col = 'P'), col = 'orange') +
  ylim(-2.2, 2.2)+
  scale_x_date(limits= as.Date(c('2000-01-01', '2003-12-31')))+
  theme_bw()+
  guides(col=guide_legend(title="Variable")) +
  theme(legend.position="bottom")
```