Example streamMetabolizer

Aline Ortega Pieck and Vanessa Garayburu-Caruso

11/5/2021

Contents

streamMetabolizer example

This guide uses data from Missouri Flat Creek, ID collected by Aline Ortega Pieck.

1. Install packages and loading libraries

If this is your first time running this code or streamMetabolizer, make sure to install the appropriate packages below. uncomment them in the chunk before Knitting.

```
#install.packages(remotes); library(remotes)
# remotes::install_github('appling/unitted', force = TRUE)
# remotes::install_github("USGS-R/streamMetabolizer", force = TRUE)
#install.packages("rstan", dependencies = TRUE)
#install.packages(devtools)

# If you have trouble installing rstan, try the installation codes below:

#devtools::install_github("stan-dev/rstan", ref = "develop", subdir = "rstan/rstan", force = TRUE)
#install.packages("rstan", type = "source")

# Run the line below if you have trouble installing devtools
#devtools::install_github("stan-dev/rstan", ref = "develop", subdir = "rstan/rstan")
```

Loading libraries

```
rm(list=ls(all=T))

library(streamMetabolizer)
library(dplyr)
library(unitted)
library(ggplot2)
library(tidyr)
library(devtools)
library(rstan)
# Correctly installing rstan can be problematic, see GitHub for issues
```

2. Setting up the data

Set working directory and read in the data and change units to match the needs of stream metabolizer. Note: Make sure to always double check that your date time column is in the date time format and not as character

```
data.path = "C:/Users/gara009/OneDrive - PNNL/Documents/01_MFC Manuscript/Data"
setwd(data.path)
 dat = read.csv("SM_WY_2020.csv",header=T)
  file.name = 'SM_WY_2020_'
 # Change date time format
dat$DATE_TIME = as.POSIXct(dat$DATE_TIME, format = "%m/%d/%Y %H:%M", tz="Etc/GMT+8")

# Transform date time into solar time
      dat$DATE_TIME = calc_solar_time(dat$DATE_TIME, longitude = -117.1)
      # Longitude is from your field site

# Change discharge from L/s to m3
      dat$Q_m3 = dat$Q_L_per_s*0.001

# Calculate in mg/L for much DO you would have in the water at the current saturation conditions
      dat$DOsat_mg_per_L =(dat$DO_mg_per_L*100)/dat$Dosat_pct
```

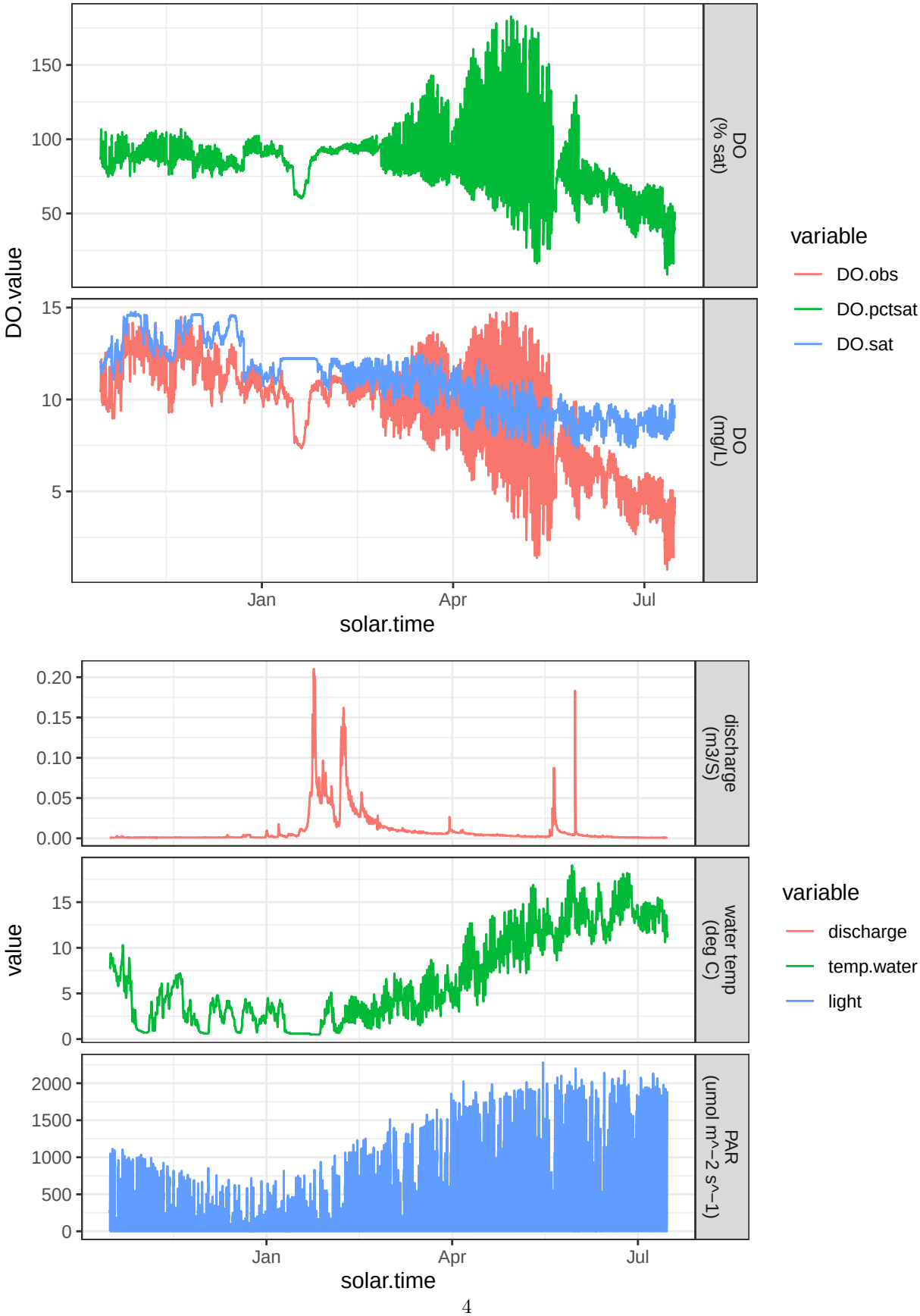
```
  # Reducing the number of decimals after performing the saturation calculations
  dat$Dosat_mg_per_L = round(dat$DOsat_mg_per_L,2)

# Selecting the data types that are needed for stream metabolizer and changing header names
      temp = dat
     dat = cbind.data.frame(temp$DATE_TIME,temp$DO_mg_per_L,
                            temp$DOsat_mg_per_L,temp$DEPTH_m,
                            temp$TEMP_degreesC,
                            temp$PAR_umol_per_m2_per_sec,temp$Q_m3)
     colnames(dat) = c("solar.time","DO.obs","DO.sat","depth","temp.water","light","discharge")
```

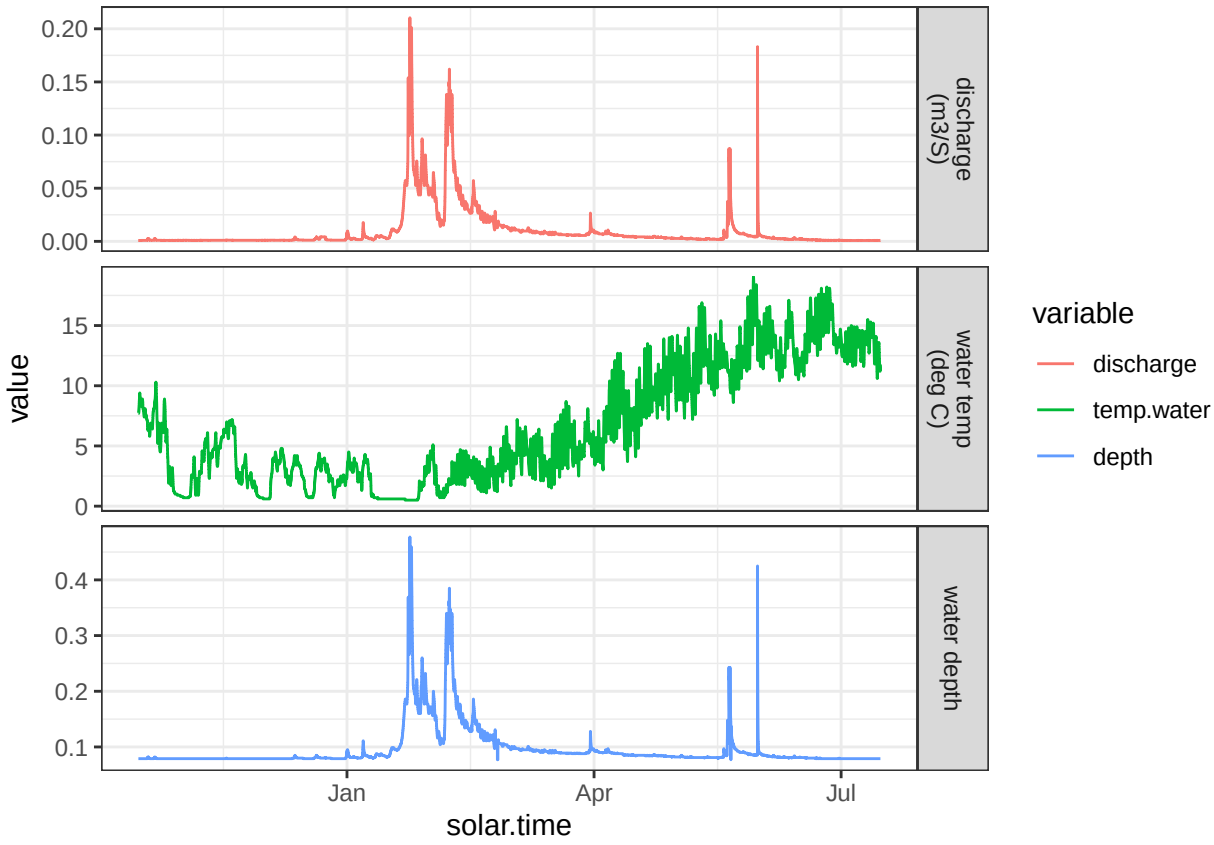
Check the number of cores you have in your computer. Based on the number that it prints, set the number of cores you want to dedicate to the metabolism run. It is recommended to set 2-4 cores less than you have in your computer to minimize the chances of R crashing. It is also recommended to select a pair number for your run.

```
parallel::detectCores()
```

```
## [1] 12
```

3. Inspect Data





4. Configure the model

We will select a Bayesian model. Then we will configure the specs of the model depending on the needs of our run.
You can play around with the number of iterations (e.g., 100 burnin iterations , also called warm up) and 50 saved steps. You can adjust the number of iterations based on the convergence of the convergence of the model.
For example, you may start with 1000 and 500 and then up the numbers to 2000 and 1000 if the model results don't seem to converge.

If you have already adjusted the number of steps multiple times and your model fits are still not good (e.g., negative GPP values) you might have to consider changing other specs in the model. Some examples of variables that you may change are: GPP_daily_lower = 0.01,ER_daily_upper = -0.01. However, you should consult the Help for more information.

Use the command plot_distribs if you want to observe the distribution of the specs if they were changed.

```
# Set the model
bayes_name = mm_name(type='bayes',
                     pool_K600='linear',
                     err_obs_iid=TRUE,
                     err_proc_iid=TRUE)
 # Options for pool K600 are binned, linear, none and normal

# Changing the specs
bayes_specs = specs(bayes_name, burnin_steps=2000,
```

```
                    saved_steps=1000, n_cores=10,
                    K600_daily_sigma_sigma = 0.2)
```

5. Fit the model and save the results

Fitting the model might take hours or days depending on the dataset. Some times R crashes while you are running the model so you want to make sure to save your results in each run. Create an output folder inside of the path where you are storing the data, the output path will update automatically here once you change your data path in step 2.

Some key parameters to look at in the mm output are:
- $daily (includes metabolism estimates) - $overall (includes error information) - $KQ_overall (included the relationship between K600-Q)

```
mm = metab(bayes_specs, data=dat)#

#Extracting the data from the model output the outputs are in a S4
#class of data and you'll need to operators to extract the daily
#time series of estimates
#get_fit(mm) %>%
  #lapply(names)

# Saving key data
output.path = paste0(data.path,"/Output/")
preds = mm@fit$daily
#str(preds)
write.csv(preds,paste0(output.path,"Results_",file.name,Sys.Date(),".csv"))

instant = mm@fit$inst
#str(instant)
write.csv(instant,paste0(output.path,"Instant_",file.name,Sys.Date(),".csv"))

Overall = mm@fit$overall
#str(Overall)
write.csv(Overall,paste0(output.path,"Overall_",file.name,Sys.Date(),".csv"))

KQ = mm@fit$KQ_overall
#str(KQ)
write.csv(KQ,paste0(output.path,"KQ_ovearll",file.name,Sys.Date(),".csv"))


#get_data(mm)# Shows a table with all the data + DO modeled
#get_data_daily(mm) #daily fitting data. It shows values for Q for now
#get_params(mm)
```

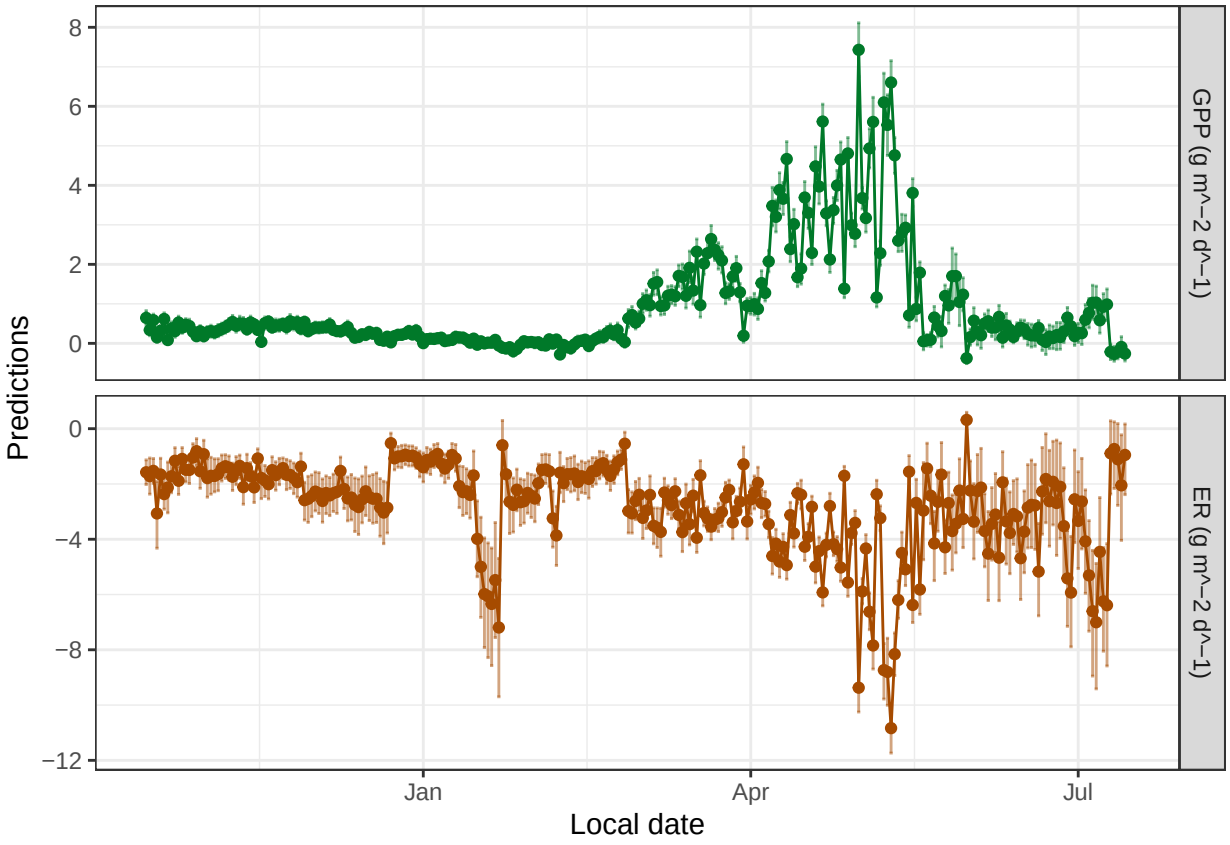
6. Inspect GPP, ER and K600

Daily predictions of modeled GPP and ER

The goal is for the predictions (lines) and observations (points) to be very similar.

```
## # A tibble: 274 x 10
```

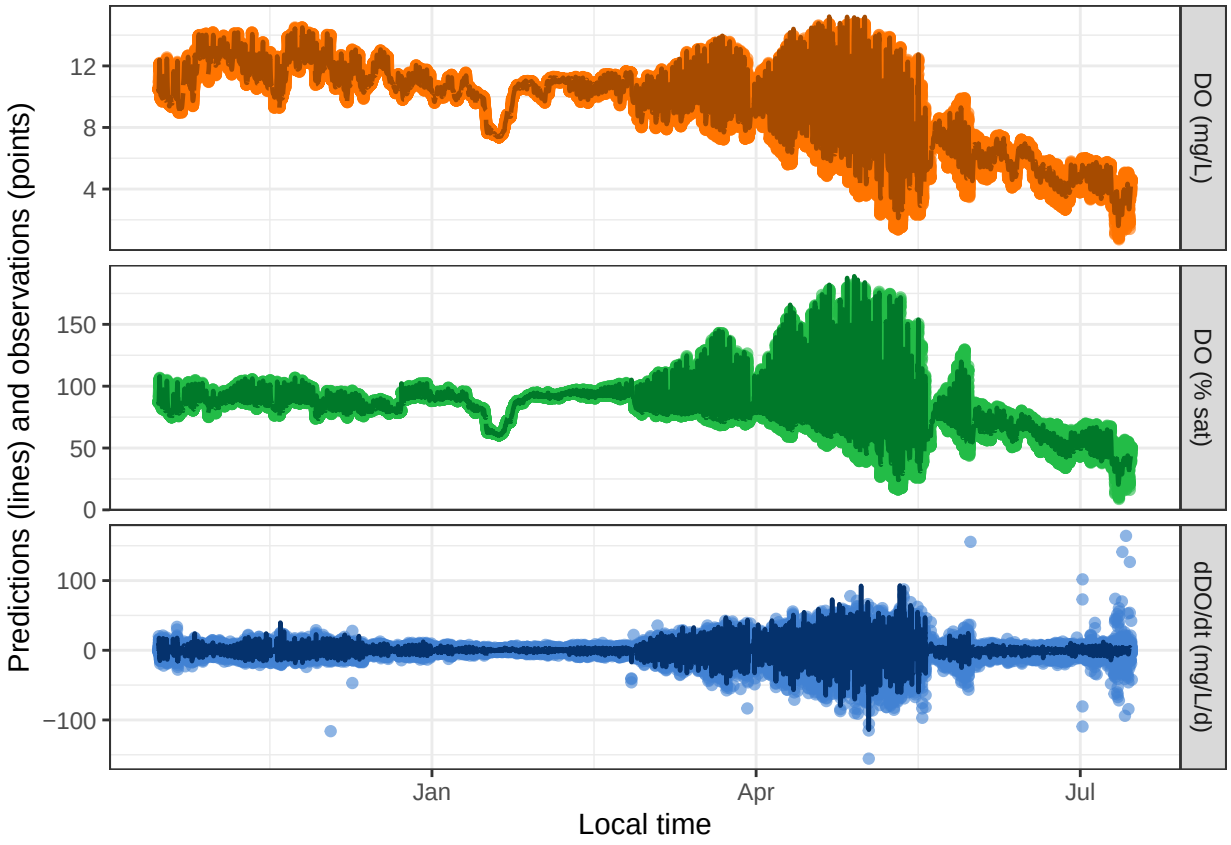
```
##    date          GPP GPP.lower GPP.upper    ER ER.lower ER.upper msgs.fit
##    <date>      <dbl>     <dbl>     <dbl> <dbl>    <dbl>    <dbl> <chr>
##  1 2019-10-16 0.645     0.468     0.828 -1.58    -2.02   -1.13  "w      "
##  2 2019-10-17 0.337     0.192     0.483 -1.72    -2.37   -1.06  "w      "
##  3 2019-10-18 0.586     0.429     0.740 -1.53    -1.98   -1.08  "w      "
##  4 2019-10-19 0.152     0.0264    0.285 -3.07    -4.32   -1.91  "w      "
##  5 2019-10-20 0.320     0.191     0.445 -1.66    -2.23   -1.07  "w      "
##  6 2019-10-21 0.617     0.455     0.772 -2.37    -2.91   -1.79  "w      "
##  7 2019-10-22 0.0838   -0.0301    0.196 -2.13    -3.04   -1.21  "w      "
##  8 2019-10-23 0.376     0.218     0.543 -1.72    -2.32   -1.15  "w      "
##  9 2019-10-24 0.308     0.153     0.462 -1.16    -1.62   -0.687 "w      "
## 10 2019-10-25 0.527     0.363     0.687 -1.88    -2.38   -1.37  "w      "
## # ... with 264 more rows, and 2 more variables: warnings <chr>, errors <chr>
```




```
## # A tibble: 274 x 9
##    date       GPP.daily GPP.daily.sd ER.daily ER.daily.sd K600.daily
##    <date>         <dbl>        <dbl>    <dbl>       <dbl>      <dbl>
##  1 2019-10-16    0.645        0.0940    -1.57       0.228       15.0
##  2 2019-10-17    0.338        0.0749    -1.72       0.331       13.7
##  3 2019-10-18    0.586        0.0795    -1.53       0.229       10.5
##  4 2019-10-19    0.153        0.0673    -3.09       0.606       17.6
##  5 2019-10-20    0.320        0.0639    -1.65       0.299       11.3
##  6 2019-10-21    0.617        0.0799    -2.36       0.292       14.5
##  7 2019-10-22    0.0841       0.0578    -2.13       0.472       16.2
##  8 2019-10-23    0.376        0.0817    -1.73       0.296       17.9
```

```
##  9 2019-10-24    0.308        0.0782    -1.16       0.234       11.7
## 10 2019-10-25    0.527        0.0831    -1.88       0.255       15.3
## # ... with 264 more rows, and 3 more variables: K600.daily.sd <dbl>,
## #   warnings <chr>, errors <chr>
```

```
##         date          solar.time DO.obs   DO.sat depth temp.water light
## 1 2019-10-16 2019-10-16 04:12:53  10.48 12.03215 0.079        7.7     0
## 2 2019-10-16 2019-10-16 04:27:53  10.47 12.03448 0.079        7.7     0
## 3 2019-10-16 2019-10-16 04:42:53  10.49 12.02982 0.079        7.7     0
## 4 2019-10-16 2019-10-16 04:57:53  10.46 12.02299 0.079        7.7     0
## 5 2019-10-16 2019-10-16 05:12:53  10.50 12.02749 0.079        7.7     0
## 6 2019-10-16 2019-10-16 05:27:53  10.49 12.01604 0.079        7.7     0
##   discharge   DO.mod
## 1     8e-04 10.48000
## 2     8e-04 10.45982
## 3     8e-04 10.44180
## 4     8e-04 10.42516
## 5     8e-04 10.41028
## 6     8e-04 10.39668
```



Ideally, good model results should have n_eff > 100 and Rhat < = 1.1. Below is a summary of these metrics for daily GPP, ER and K600.

```
## # A tibble: 274 x 7
##    GPP_Rhat ER_Rhat DO_R2_Rhat GPP_daily_Rhat ER_daily_Rhat K600_daily_Rhat
##       <dbl>   <dbl>      <dbl>          <dbl>         <dbl>           <dbl>
```

```
##  1     1.01    1.01       2.30           1.01          1.01            1.01
##  2     1.01    1.01       2.26           1.01          1.01            1.01
##  3     1.01    1.01       2.27           1.01          1.01            1.01
##  4     1.00    1.01       2.34           1.00          1.01            1.01
##  5     1.01    1.01       2.31           1.01          1.01            1.00
##  6     1.01    1.01       2.26           1.01          1.01            1.01
##  7     1.01    1.01       2.28           1.01          1.01            1.01
##  8     1.01    1.01       2.29           1.01          1.01            1.01
##  9     1.00    1.01       2.33           1.00          1.01            1.01
## 10     1.01    1.01       2.29           1.01          1.01            1.01
## # ... with 264 more rows, and 1 more variable: K600_daily_predlog_Rhat <dbl>
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   1.000   1.005   1.008   1.009   1.011   1.038       1
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  0.9996  1.0036  1.0056  1.0070  1.0089  1.0299       1
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  0.9998  1.0044  1.0069  1.0083  1.0107  1.0346       1
```

```
## # A tibble: 274 x 7
##    GPP_n_eff ER_n_eff DO_R2_n_eff GPP_daily_n_eff ER_daily_n_eff
##        <dbl>    <dbl>       <dbl>           <dbl>          <dbl>
##  1      483.     448.        3.04            483.           448.
##  2      518.     382.        2.96            518.           382.
##  3      412.     337.        3.02            412.           337.
##  4      728.     480.        2.94            728.           480.
##  5      636.     494.        2.97            636.           494.
##  6      428.     374.        3.05            428.           374.
##  7      650.     356.        3.00            650.           356.
##  8      458.     403.        3.02            458.           403.
##  9      479.     418.        2.94            479.           418.
## 10      473.     426.        2.99            473.           426.
## # ... with 264 more rows, and 2 more variables: K600_daily_n_eff <dbl>,
## #   K600_daily_predlog_n_eff <dbl>
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   227.0   361.4   409.4   412.4   458.9   694.9       1
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   243.8   386.5   470.9   546.3   651.8  1267.1       1
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   221.7   367.3   426.2   427.2   480.2   696.8       1
```

7. Inspect errors and their standard deviations

err_obs_iid_Rhat should have a value < 1.1. Additionally, in a conversation with Allison Appling she mentioned that they have seen pretty frequently err_obs_iid_sigma_Rhats much greater than 1.05, and mentioned it on the JGR 2018 paper. This is OK for that particular parameter because the values that the model continues to consider are usually quite similar in absolute magnitudes.

Inspect err__obs__iid__Rhat and err__obs__iid__sigma__Rhat

```
#err_obs_iid_Rhat
get_fit(mm)$inst %>%
  select(ends_with('Rhat'))
```

```
## # A tibble: 26,208 x 2
##    err_obs_iid_Rhat err_proc_iid_Rhat
##               <dbl>             <dbl>
##  1            0.999              1.00
##  2            1.00               1.00
##  3            1.00               1.00
##  4            1.00               1.00
##  5            1.00               1.00
##  6            0.999              1.00
##  7            0.999              1.00
##  8            1.00               1.00
##  9            1.00               1.00
## 10            1.00               1.00
## # ... with 26,198 more rows
```

```
# err_obs_iid_sigma_Rhat. See comment from Appling
get_fit(mm)$overall %>%
  select(ends_with('Rhat'))
```

```
## # A tibble: 1 x 3
##   err_obs_iid_sigma_Rhat err_proc_iid_sigma_Rhat lp___Rhat
##                    <dbl>                   <dbl>     <dbl>
## 1                   3.33                    1.00      3.19
```

```
get_fit(mm)$inst %>%
  select(ends_with('n_eff'))
```

```
## # A tibble: 26,208 x 2
##    err_obs_iid_n_eff err_proc_iid_n_eff
##                <dbl>              <dbl>
##  1             5423.              3441.
##  2             5353.              3265.
##  3             5649.              3454.
##  4             6866.              3161.
##  5             5386.              2985.
##  6             5702.              3012.
##  7             6092.              3491.
##  8             6375.              3562.
##  9             6118.              3714.
## 10             5472.              4278.
## # ... with 26,198 more rows
```

```
get_fit(mm)$overall %>%
  select(ends_with('n_eff'))
```

```
## # A tibble: 1 x 3
##   err_obs_iid_sigma_n_eff err_proc_iid_sigma_n_eff lp___n_eff
##                     <dbl>                    <dbl>      <dbl>
## 1                    2.85                    1087.       3.14
```

```
summary(instant$err_obs_iid_n_eff)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1411    5537    5880    5895    6239    8295
```

```
summary(instant$err_obs_iid_Rhat)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.9990  0.9994  0.9996  0.9997  0.9999  1.0105
```

```
summary(instant$err_proc_iid_n_eff)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   245.2  1323.6  2449.5  2808.4  3922.3  9406.3     273
```

```
summary(instant$err_proc_iid_Rhat)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  0.9990  0.9999  1.0006  1.0014  1.0020  1.0370     273
```

```
summary(Overall$err_obs_iid_sigma_Rhat)# There is only one err_obs_iid_sigma_Rhat per run
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.332   3.332   3.332   3.332   3.332   3.332
```

```
summary(Overall$err_proc_iid_sigma_Rhat)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.002   1.002   1.002   1.002   1.002   1.002
```

```
summary(Overall$err_obs_iid_sigma_n_eff)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.848   2.848   2.848   2.848   2.848   2.848
```

```
summary(Overall$err_proc_iid_sigma_n_eff)
```

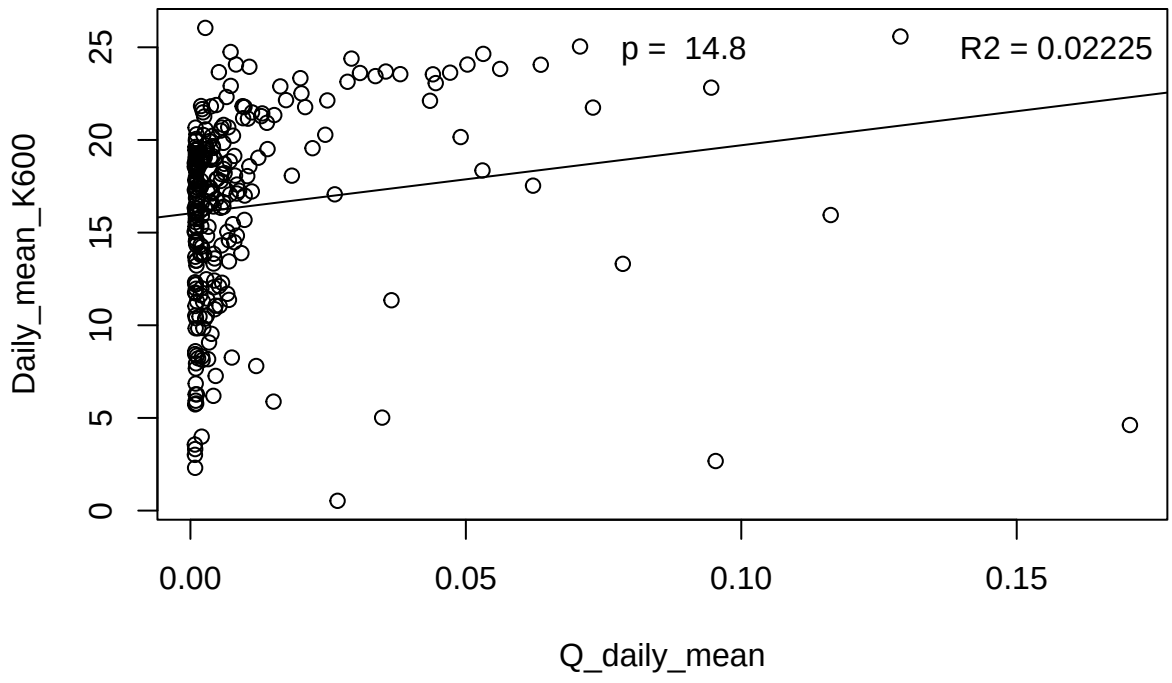
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1087    1087    1087    1087    1087    1087
```

8. Inspect the relationships between variables

Relationship between K600 and Q

The first step is to plot K600 and Q, this relationship should be linear. To find this relationship we have to calculate the mean Q per day and then plot against K600.

```
    dat$solar.time = as.Date(dat$solar.time)
    meanQday = aggregate(dat["discharge"],by = dat["solar.time"],mean)
    K600Q = lm(preds$K600_daily_mean~meanQday$discharge)

    plot(meanQday$discharge,preds$K600_daily_mean,
         xlab="Q_daily_mean",ylab="Daily_mean_K600")
    abline(lm(preds$K600_daily_mean~meanQday$discharge))
      legend("topright", bty="n", legend=paste("R2 =", format(summary(K600Q)$r.squared, digits=4)))
legend("top", bty="n", legend=paste("p = ", format(summary(K600Q)$coefficients[4], digits=4)))
```




```
    summary(K600Q)
```

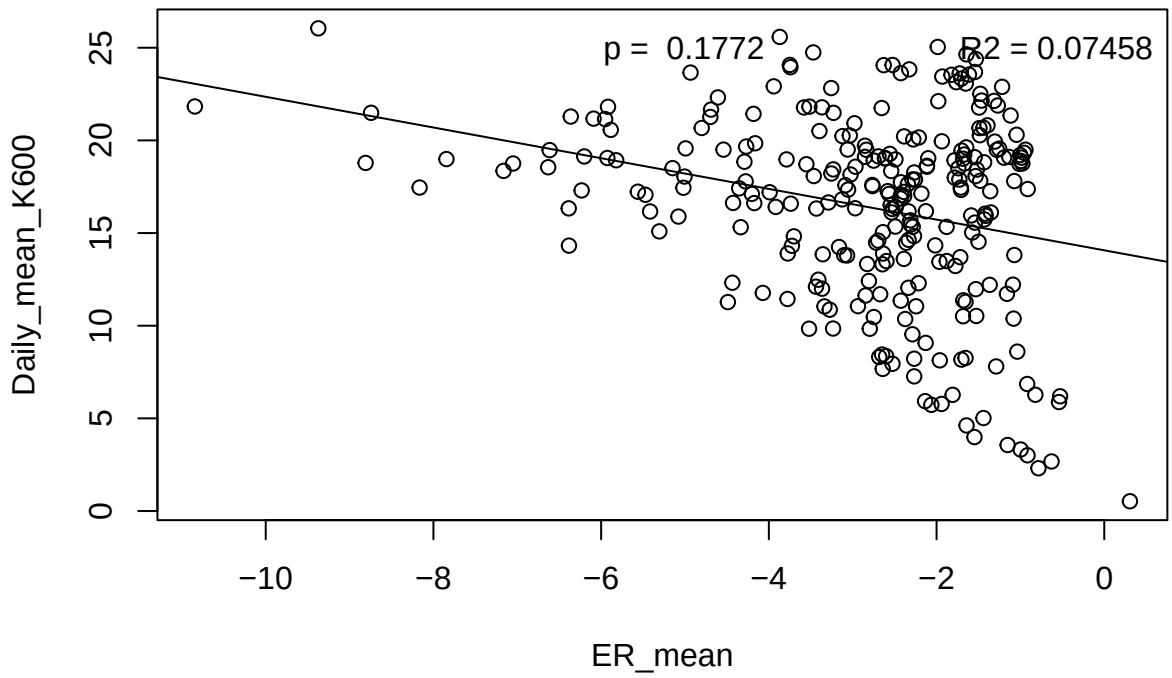
```
##
## Call:
## lm(formula = preds$K600_daily_mean ~ meanQday$discharge)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -17.689  -2.590   1.245   3.344   9.909
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)          16.038      0.339  47.309   <2e-16 ***
## meanQday$discharge   36.752     14.799   2.484   0.0136 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.032 on 271 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.02225,    Adjusted R-squared:  0.01865
## F-statistic: 6.168 on 1 and 271 DF,  p-value: 0.01361
```

Relationship between K600 and ER

This relationship is key. Bob Hall says: My primary metric is to look for covariance between ER and K. These have high equifinality which means that any combination of ER or K can provide an equally good fit to the model. If these covary strongly (r=0.6 or higher) then that may not be good for examining controls on variation in ER. If you find **no relationship**, that is a **very good** thing. If there is a relationship, that is if R2 is high, it means that the model can't parse reaeration from respiration. In this case, ER and K600 are working in opposite for O2, so ideally there is no relationship in the model outputs.

```
K600ER = lm(K600_daily_mean~ER_mean, data=preds)

    plot(preds$ER_mean,preds$K600_daily_mean,xlab="ER_mean",ylab="Daily_mean_K600")
    abline(lm(K600_daily_mean~ER_mean, data=preds))
    legend("topright", bty="n", legend=paste("R2 =", format(summary(K600ER)$r.squared, digits=4)))
legend("top", bty="n", legend=paste("p = ", format(summary(K600ER)$coefficients[4], digits=4)))
```



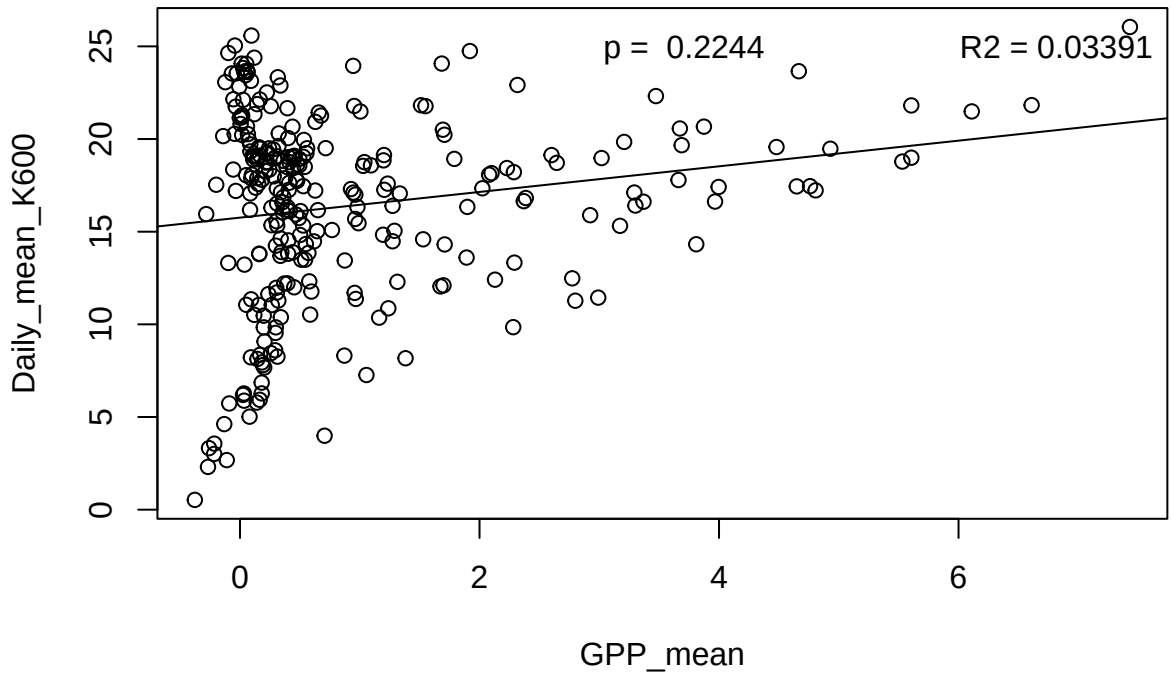

```
summary(K600ER)
```

```
##
## Call:
## lm(formula = K600_daily_mean ~ ER_mean, data = preds)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.2858  -2.9455   0.4828   3.5631   9.3291
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  14.0684     0.5816  24.189  < 2e-16 ***
## ER_mean      -0.8283     0.1772  -4.673 4.67e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.896 on 271 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.07458,    Adjusted R-squared:  0.07117
## F-statistic: 21.84 on 1 and 271 DF,  p-value: 4.674e-06
```

Relationship between K600 and GPP

High GPP is needed to estimate K well, so K should get more variable as GPP decreases.

```
K600GPP = lm(K600_daily_mean~GPP_mean, data=preds)


    plot(preds$GPP_mean,preds$K600_daily_mean,xlab="GPP_mean",ylab="Daily_mean_K600")
    abline(lm(K600_daily_mean~GPP_mean, data=preds))
    legend("topright", bty="n", legend=paste("R2 =", format(summary(K600GPP)$r.squared, digits=4)))
legend("top", bty="n", legend=paste("p = ", format(summary(K600GPP)$coefficients[4], digits=4)))
```




```
    summary(K600GPP)
```

```
##
## Call:
## lm(formula = K600_daily_mean ~ GPP_mean, data = preds)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.9694  -2.6299   0.8774   3.2720   9.7564
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  15.7593     0.3685  42.766  < 2e-16 ***
## GPP_mean      0.6921     0.2244   3.084  0.00225 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 5.002 on 271 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.03391,    Adjusted R-squared:  0.03035
## F-statistic: 9.513 on 1 and 271 DF,  p-value: 0.002252
```

Time series daily mean K600

```
    preds$date = as.Date(preds$date)

    plot(preds$date,preds$K600_daily_mean,
         xlab="Date",ylab="Daily_mean_K600")
```