Prediction

Edward J. Xu

2019-09-15

```
library(tidyverse)
library(tidymodels)
library(magrittr)
library(ggplot2)
library(lubridate)
library(stringr)
library(splines)
library(glue)

files <- dir(
  "~/GitHub/tidynamics/vignettes/funcs/pred",
  full.names = TRUE
  )
for (i in 1:length(files)) {
  source(files[i])
}
```

1, Data Process

```
## The data is stored in list
li <- readRDS("~/GitHub/tidynamics/data/soenderborg.RDS")

## Change the column names in `li$Gnwp` like "k1" to "t1"
for (i in names(li$Gnwp)) {
  li$Gnwp[paste0("t", str_sub(i, 2, -1))] = li$Gnwp[i]
}
```

2, Relate Tibbles

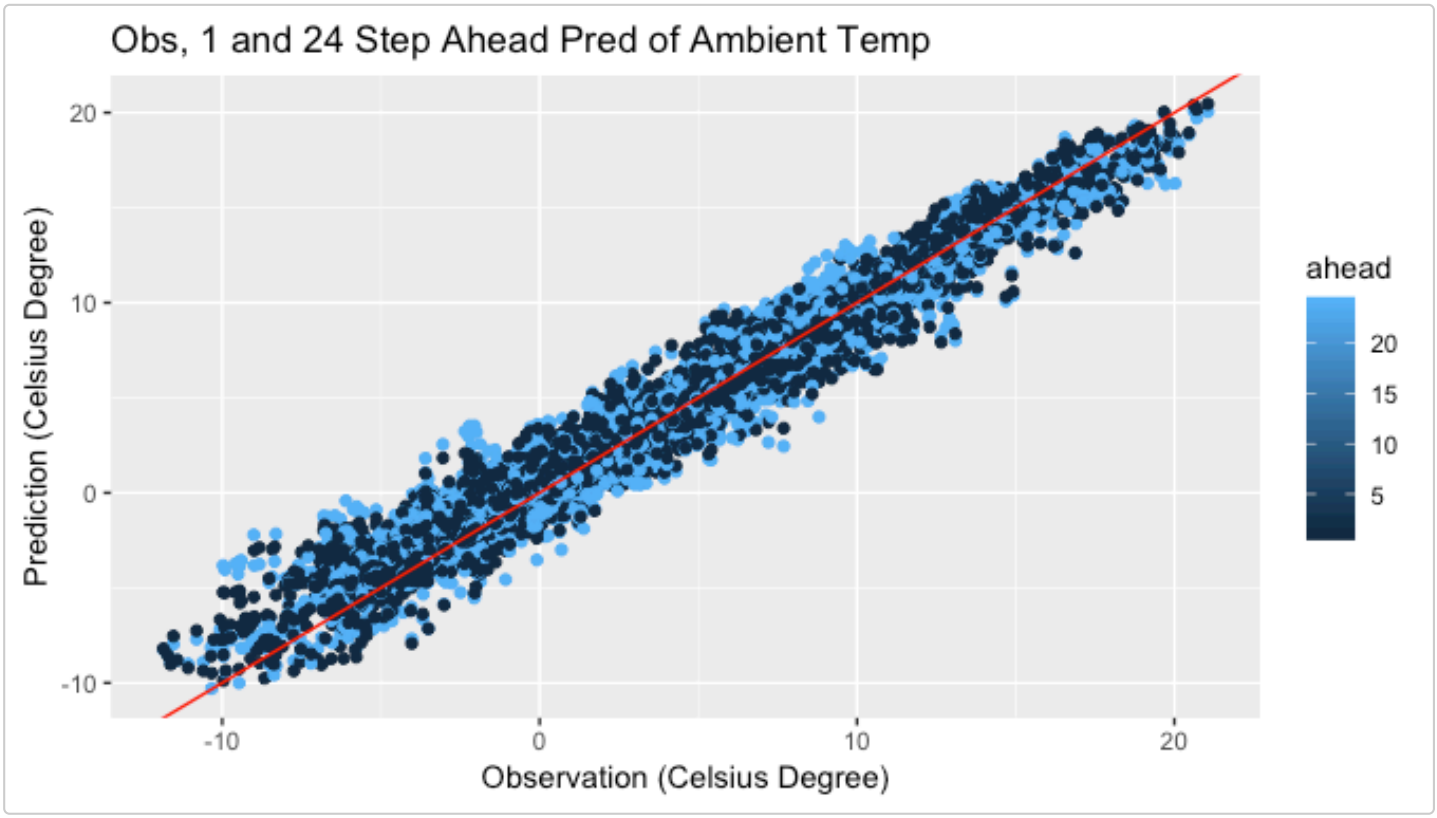
```
#' Get the first character in the string
str_sub_1 <- function(chr){
  i <- NA
  if (str_sub(chr, 1, 1) == "k") {
    i <- "t_a.p"
  } else {
    i <- "g.p"
  }
  return(i)
}
```

```r
#' Get the value of step ahead
get_ahead <- function(chr){
  return(strtoi(str_sub(chr, 2, -1)))
}

ti <- as_tibble(
    cbind(
      data.frame(
        "time" = li$t, "t_a" = li$Ta, "g" = li$G), li$Tanwp, li$Gnwp[, 50:98],
        "ph1" = li$Ph1, "ph2" = li$Ph2, "ph3" = li$Ph3
    )
  )

rm(li)

ti %>%  # Check if `time` is the primary key
  count(time) %>%
  {nrow(filter(., n > 1)) == 0}
#> [1] TRUE

li <- list()

li$time <- ti %>%
  mutate(at = 1:nrow(.)) %>%
  mutate(fo = 1:nrow(.)) %>%
  dplyr::select(at, fo, time)

li$obs <- ti %>%
  mutate(fo = 1:nrow(.)) %>%
  dplyr::select(fo, t_a, g, ph1, ph2, ph3)

li$pred <- ti %>%
  mutate(at = 1:nrow(.)) %>%
  dplyr::select(at, 4:90) %>%
  gather(-at, key = "ahead_chr", value = "pred") %>%
  mutate(whi = map_chr(ahead_chr, str_sub_1)) %>%
  mutate(ahead = map_int(ahead_chr, get_ahead)) %>%
  dplyr::select(-ahead_chr) %>%
  mutate(fo = at + ahead) %>%
  spread(key = whi, value = pred) %>%
  arrange(at)

# saveRDS(li, "~GitHub/tidynamics/data/soenderborg_tidy.RDS")
```

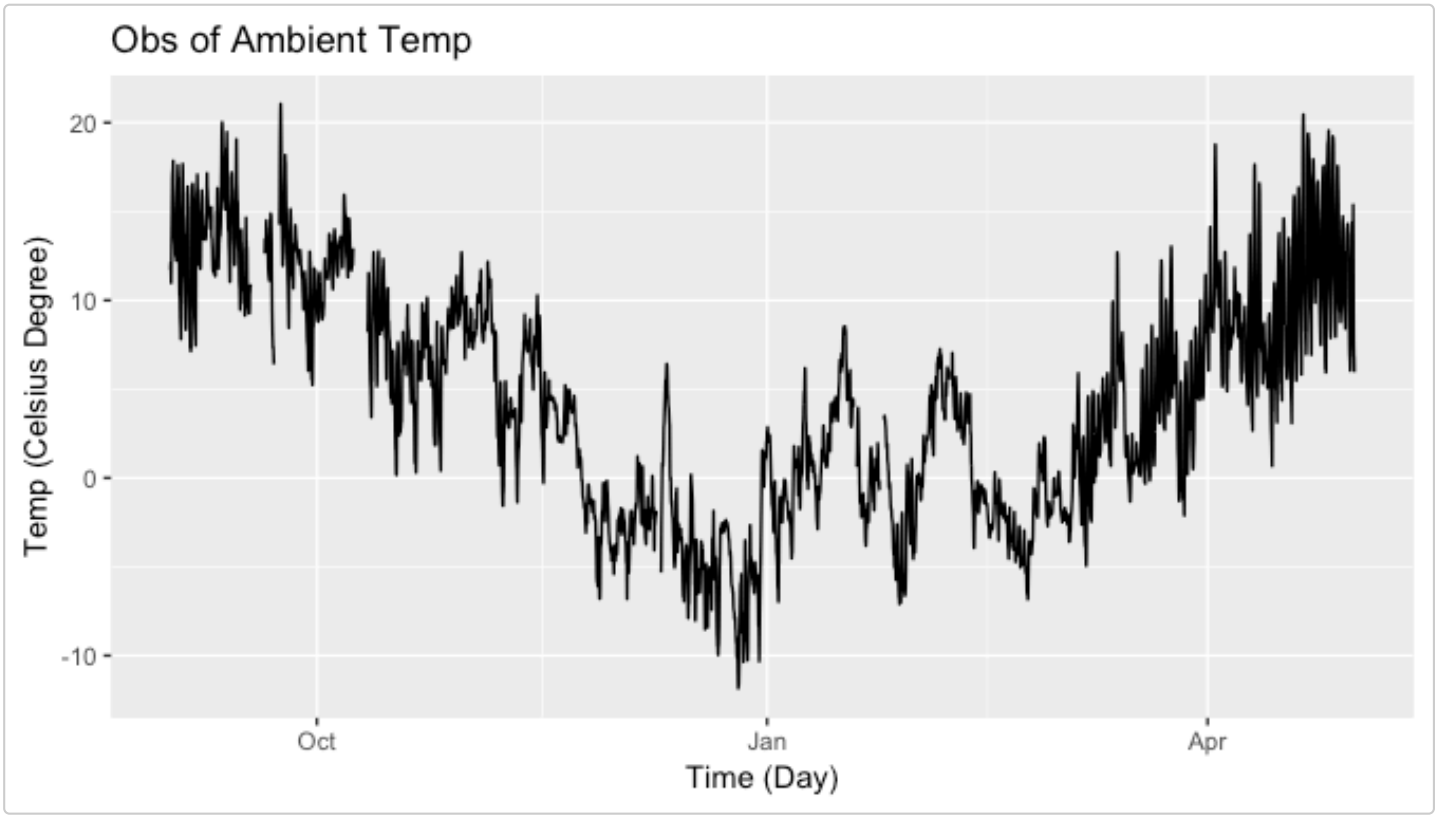
3, Visualize Forecast

```r
li$pred %>%
  left_join(li$obs, by = "fo") %>%
  filter(ahead %in% c(1, 24)) %>%
  ggplot() +
    geom_point(mapping = aes(x = t_a, y = t_a.p, color = ahead)) +
    geom_abline(aes(intercept = 0, slope = 1), color = "red") +
```

```
    labs(
      title = "Obs, 1 and 24 Step Ahead Pred of Ambient Temp",
      x = "Observation (Celsius Degree)", y = "Prediction (Celsius Degree)"
      )
```




```
li$obs %>%
  left_join(li$time, by = "fo") %>%
  ggplot() +
  geom_line(mapping = aes(x = time, y = t_a)) +
  labs(
    title = "Obs of Ambient Temp",
    x = "Time (Day)", y = "Temp (Celsius Degree)"
  )
```



4, Functions for Tuning

```r
#' Get recipe from split
get_li_dm_1 <- function(split, coef_t, coef_g) {
  rec <-
    training(split) %>%
    recipe(ph3 ~ t_a.p + g.p) %>%
    step_mutate(t_a.p.lp = lp_vector(t_a.p, a1 = coef_t)) %>%
    step_mutate(g.p.lp = lp_vector(g.p, a1 = coef_g)) %>%
    # step_corr(all_predictors()) %>%
    step_center(all_predictors(), -all_outcomes()) %>%
    step_scale(all_predictors(), -all_outcomes()) %>%
    prep()

  dm_train <- juice(rec)
  dm_test <- bake(rec, testing(split))

  return(list("train" = dm_train, "test" = dm_test, "rec" = rec))
}

# Get the model from `ti_train`
get_mod <- function(ti_train = li_dm$train) {
  mod_lm <- linear_reg() %>%
    set_engine("lm") %>%
    fit(ph3 ~ ., data = ti_train)
  return(mod_lm)
}

#' Evaluate the training and testing design matrix
get_rmse <- function(mod_lm, li_dm) {
```

```
  rmse <-
    li_dm$test %>%
    bind_cols(predict(mod_lm, .)) %>%
    metrics(truth = ph3, estimate = .pred) %>%
    `[[`(1, 3)

  return(rmse)
}

#' Evaluate the set of parameters for linear regression
val_para <- function(split, coef_t, coef_g, func_li_dm = get_li_dm_1) {
  li_dm <-
    split %>%
    func_li_dm(coef_t, coef_g)

  rmse <-
    li_dm %>%
    {get_mod(.$train)} %>%
    get_rmse(li_dm)

  return(rmse)
}

#' To cross validate the parameters
cv_para <- function(para, ti_cv, func_li_dm = get_li_dm_1) {
  rmse_mean <-
    ti_cv$splits %>%  # All the splits for cross validation
    # unlist(use.names = FALSE) %>%
    map_dbl(
      val_para, coef_t = para[1], coef_g = para[2], func_li_dm = func_li_dm
    ) %>%  # Apply the val_para to split one by one
    mean()

  rmse_mean %>%
    {glue("
      coef_t = {para[1]} ; coef_g = {para[2]} ; rmse_mean = {.} ;
      ")} %>%
    message()

  return(rmse_mean)
}
```

```
## Split to training and testing sets
## Select the heating load from house 3
split_a1 <- li$pred %>%
  filter(ahead == 1) %>%
  left_join(li$time, by = "fo") %>%
  left_join(li$obs, by = "fo") %>%
  mutate(hour = as.numeric(hour(.$time))) %>%
  dplyr::select(fo, hour, t_a.p, g.p, ph3) %>%
  initial_split(0.6)

ti_cv <-
```

```
  split_a1 %>%
  training() %>%
  vfold_cv(v = 10, repeats = 1)

## Test the `get_li_dm_1` function
ti_cv %>%
  {.$splits[[1]]} %>%
  {get_li_dm_1(., 0.9, 0.9)} %>%
  print()
#> $train
#> # A tibble: 3,136 x 5
#>    t_a.p    g.p   ph3 t_a.p.lp g.p.lp
#>    <dbl>  <dbl> <dbl>    <dbl>  <dbl>
#>  1  1.27 -0.558 0.833     1.35 -1.15
#>  2  1.21 -0.558 0.583     1.34 -1.15
#>  3  1.16 -0.558 0.550     1.33 -1.15
#>  4  1.33  0.397 0.333     1.34 -0.953
#>  5  1.78  0.948 0.417     1.39 -0.658
#>  6  1.70  0.248 4.67      1.44 -0.539
#>  7  1.41 -0.558 0.317     1.44 -0.600
#>  8  1.21 -0.558 0.633     1.43 -0.655
#>  9  1.16 -0.558 0.400     1.41 -0.705
#> 10  1.08 -0.558 1.10      1.38 -0.750
#> # … with 3,126 more rows
#>
#> $test
#> # A tibble: 349 x 5
#>    t_a.p      g.p   ph3 t_a.p.lp  g.p.lp
#>    <dbl>    <dbl> <dbl>    <dbl>   <dbl>
#>  1 1.21  -0.0255  0.467     1.28 -0.0401
#>  2 1.89  -0.00314 0.400     1.35 -0.0355
#>  3 1.12  -0.119   0.283     1.34 -0.0554
#>  4 1.67   2.52    0.567     1.38  0.478
#>  5 0.750  0.118   0.533     1.32  0.456
#>  6 0.730 -0.558   0.383     1.27  0.296
#>  7 1.62   0.527   0.467     1.31  0.377
#>  8 1.24  -0.558   0.350     1.31  0.224
#>  9 1.37  -0.246   0.767     1.33  0.152
#> 10 1.63   2.20    0.717     1.37  0.598
#> # … with 339 more rows
#>
#> $rec
#> Data Recipe
#>
#> Inputs:
#>
#>       role #variables
#>    outcome          1
#>  predictor          2
#>
#> Training data contained 3136 data points and 193 incomplete rows.
#>
#> Operations:
#>
```

```
#> Variable mutation for  t_a.p.lp [trained]
#> Variable mutation for  g.p.lp [trained]
#> Centering for t_a.p, g.p, t_a.p.lp, g.p.lp [trained]
#> Scaling for t_a.p, g.p, t_a.p.lp, g.p.lp [trained]

##
rec <-
  ti_cv %>%
  {.$splits[[1]]} %>%
  {get_li_dm_1(., 0.9, 0.9)} %>%
  {.$rec}

## Test the `cv_para` function
c(0.9, 0.9) %>%
  cv_para(ti_cv, get_li_dm_1)
#> coef_t = 0.9 ; coef_g = 0.9 ; rmse_mean = 1.00961949155428 ;
#> [1] 1.009619
```

5, Tune the Low-Pass Coeff

```
## Optimize the choice of low-pass filtering coefficients
result <-
  optim(
    c(t = 0.98, g = 0.98), cv_para,
    lower = c(0.3, 0.1), upper = c(0.999, 0.999), method = "L-BFGS-B",
    ti_cv = ti_cv, func_li_dm = get_li_dm_1
    ) %>%
  print()
#> coef_t = 0.98 ; coef_g = 0.98 ; rmse_mean = 1.07216318531944 ;
#> coef_t = 0.981 ; coef_g = 0.98 ; rmse_mean = 1.07457257700946 ;
#> coef_t = 0.979 ; coef_g = 0.98 ; rmse_mean = 1.06981060353179 ;
#> coef_t = 0.98 ; coef_g = 0.981 ; rmse_mean = 1.07326212184901 ;
#> coef_t = 0.98 ; coef_g = 0.979 ; rmse_mean = 1.07111833116161 ;
#> coef_t = 0.3 ; coef_g = 0.1 ; rmse_mean = 1.04571945763351 ;
#> coef_t = 0.301 ; coef_g = 0.1 ; rmse_mean = 1.04569556145334 ;
#> coef_t = 0.3 ; coef_g = 0.1 ; rmse_mean = 1.04571945763351 ;
#> coef_t = 0.3 ; coef_g = 0.101 ; rmse_mean = 1.04573332143051 ;
#> coef_t = 0.3 ; coef_g = 0.1 ; rmse_mean = 1.04571945763351 ;
#> coef_t = 0.306068612821341 ; coef_g = 0.1 ; rmse_mean = 1.04557359100473 ;
#> coef_t = 0.307068612821341 ; coef_g = 0.1 ; rmse_mean = 1.04554935791001 ;
#> coef_t = 0.305068612821341 ; coef_g = 0.1 ; rmse_mean = 1.04559776803555 ;
#> coef_t = 0.306068612821341 ; coef_g = 0.101 ; rmse_mean = 1.045587177869 ;
#> coef_t = 0.306068612821341 ; coef_g = 0.1 ; rmse_mean = 1.04557359100473 ;
#> coef_t = 0.330343064106705 ; coef_g = 0.1 ; rmse_mean = 1.04496895751653 ;
#> coef_t = 0.331343064106705 ; coef_g = 0.1 ; rmse_mean = 1.04494329112161 ;
#> coef_t = 0.329343064106705 ; coef_g = 0.1 ; rmse_mean = 1.04499456193138 ;
#> coef_t = 0.330343064106705 ; coef_g = 0.101 ; rmse_mean = 1.04498140585463 ;
#> coef_t = 0.330343064106705 ; coef_g = 0.1 ; rmse_mean = 1.04496895751653 ;
#> coef_t = 0.42744086924816 ; coef_g = 0.1 ; rmse_mean = 1.04213985012145 ;
#> coef_t = 0.42844086924816 ; coef_g = 0.1 ; rmse_mean = 1.04210660683947 ;
#> coef_t = 0.42644086924816 ; coef_g = 0.1 ; rmse_mean = 1.04217299718452 ;
#> coef_t = 0.42744086924816 ; coef_g = 0.101 ; rmse_mean = 1.04214751399589 ;
```

```
#> coef_t = 0.42744086924816 ; coef_g = 0.1 ; rmse_mean = 1.04213985012145 ;
#> coef_t = 0.815832089813982 ; coef_g = 0.1 ; rmse_mean = 1.02248133523667 ;
#> coef_t = 0.816832089813982 ; coef_g = 0.1 ; rmse_mean = 1.02245149679224 ;
#> coef_t = 0.814832089813982 ; coef_g = 0.1 ; rmse_mean = 1.02251188641974 ;
#> coef_t = 0.815832089813982 ; coef_g = 0.101 ; rmse_mean = 1.02248306098443 ;
#> coef_t = 0.815832089813982 ; coef_g = 0.1 ; rmse_mean = 1.02248133523667 ;
#> coef_t = 0.999 ; coef_g = 0.1 ; rmse_mean = 1.03128740285905 ;
#> coef_t = 0.999 ; coef_g = 0.1 ; rmse_mean = 1.03128740285905 ;
#> coef_t = 0.998 ; coef_g = 0.1 ; rmse_mean = 1.04255843343289 ;
#> coef_t = 0.999 ; coef_g = 0.101 ; rmse_mean = 1.03127055677054 ;
#> coef_t = 0.999 ; coef_g = 0.1 ; rmse_mean = 1.03128740285905 ;
#> coef_t = 0.999 ; coef_g = 0.111684120225782 ; rmse_mean = 1.03108889757702 ;
#> coef_t = 0.999 ; coef_g = 0.111684120225782 ; rmse_mean = 1.03108889757702 ;
#> coef_t = 0.998 ; coef_g = 0.111684120225782 ; rmse_mean = 1.04241145026454 ;
#> coef_t = 0.999 ; coef_g = 0.112684120225782 ; rmse_mean = 1.03107174090388 ;
#> coef_t = 0.999 ; coef_g = 0.110684120225782 ; rmse_mean = 1.03110602825392 ;
#> coef_t = 0.999 ; coef_g = 0.158420601128909 ; rmse_mean = 1.03026121665357 ;
#> coef_t = 0.999 ; coef_g = 0.158420601128909 ; rmse_mean = 1.03026121665357 ;
#> coef_t = 0.998 ; coef_g = 0.158420601128909 ; rmse_mean = 1.04181983874731 ;
#> coef_t = 0.999 ; coef_g = 0.159420601128909 ; rmse_mean = 1.03024297406983 ;
#> coef_t = 0.999 ; coef_g = 0.157420601128909 ; rmse_mean = 1.03027943876075 ;
#> coef_t = 0.999 ; coef_g = 0.345366524741419 ; rmse_mean = 1.02664887956699 ;
#> coef_t = 0.999 ; coef_g = 0.345366524741419 ; rmse_mean = 1.02664887956699 ;
#> coef_t = 0.998 ; coef_g = 0.345366524741419 ; rmse_mean = 1.03975096848092 ;
#> coef_t = 0.999 ; coef_g = 0.346366524741419 ; rmse_mean = 1.02662939449942 ;
#> coef_t = 0.999 ; coef_g = 0.344366524741419 ; rmse_mean = 1.02666837393484 ;
#> coef_t = 0.999 ; coef_g = 0.999 ; rmse_mean = 1.13244585752874 ;
#> coef_t = 0.999 ; coef_g = 0.999 ; rmse_mean = 1.13244585752874 ;
#> coef_t = 0.998 ; coef_g = 0.999 ; rmse_mean = 1.15917483485166 ;
#> coef_t = 0.999 ; coef_g = 0.999 ; rmse_mean = 1.13244585752873 ;
#> coef_t = 0.999 ; coef_g = 0.998 ; rmse_mean = 1.09082649537407 ;
#> coef_t = 0.999 ; coef_g = 0.579926134807654 ; rmse_mean = 1.02290019064111 ;
#> coef_t = 0.999 ; coef_g = 0.579926134807654 ; rmse_mean = 1.02290019064111 ;
#> coef_t = 0.998 ; coef_g = 0.579926134807654 ; rmse_mean = 1.04005561603635 ;
#> coef_t = 0.999 ; coef_g = 0.580926134807654 ; rmse_mean = 1.02289129415117 ;
#> coef_t = 0.999 ; coef_g = 0.578926134807654 ; rmse_mean = 1.02290919489544 ;
#> coef_t = 0.999 ; coef_g = 0.999 ; rmse_mean = 1.13244585752873 ;
#> coef_t = 0.999 ; coef_g = 0.999 ; rmse_mean = 1.13244585752873 ;
#> coef_t = 0.998 ; coef_g = 0.999 ; rmse_mean = 1.15917483485164 ;
#> coef_t = 0.999 ; coef_g = 0.999 ; rmse_mean = 1.13244585752873 ;
#> coef_t = 0.999 ; coef_g = 0.998 ; rmse_mean = 1.09082649537407 ;
#> coef_t = 0.999 ; coef_g = 0.722189778448085 ; rmse_mean = 1.02352659854752 ;
#> coef_t = 0.999 ; coef_g = 0.722189778448085 ; rmse_mean = 1.02352659854752 ;
#> coef_t = 0.998 ; coef_g = 0.722189778448085 ; rmse_mean = 1.04418357212906 ;
#> coef_t = 0.999 ; coef_g = 0.723189778448085 ; rmse_mean = 1.02355348345448 ;
#> coef_t = 0.999 ; coef_g = 0.721189778448085 ; rmse_mean = 1.02350021090681 ;
#> coef_t = 0.999 ; coef_g = 0.634745366017947 ; rmse_mean = 1.02260492528446 ;
#> coef_t = 0.999 ; coef_g = 0.634745366017947 ; rmse_mean = 1.02260492528446 ;
#> coef_t = 0.998 ; coef_g = 0.634745366017947 ; rmse_mean = 1.04112998505098 ;
#> coef_t = 0.999 ; coef_g = 0.635745366017947 ; rmse_mean = 1.02260392249363 ;
#> coef_t = 0.999 ; coef_g = 0.633745366017947 ; rmse_mean = 1.02260611581866 ;
#> coef_t = 0.999 ; coef_g = 0.635842028532746 ; rmse_mean = 1.02260383558715 ;
#> coef_t = 0.999 ; coef_g = 0.635842028532746 ; rmse_mean = 1.02260383558715 ;
#> coef_t = 0.998 ; coef_g = 0.635842028532746 ; rmse_mean = 1.04115725167665 ;
```

```
#> coef_t = 0.999 ; coef_g = 0.636842028532746 ; rmse_mean = 1.02260304101728 ;
#> coef_t = 0.999 ; coef_g = 0.634842028532746 ; rmse_mean = 1.02260482012327 ;
#> coef_t = 0.999 ; coef_g = 0.640552287214842 ; rmse_mean = 1.02260178735905 ;
#> coef_t = 0.999 ; coef_g = 0.640552287214842 ; rmse_mean = 1.02260178735905 ;
#> coef_t = 0.998 ; coef_g = 0.640552287214842 ; rmse_mean = 1.04127717959231 ;
#> coef_t = 0.999 ; coef_g = 0.641552287214842 ; rmse_mean = 1.0226019156313 ;
#> coef_t = 0.999 ; coef_g = 0.639552287214842 ; rmse_mean = 1.0226018589546 ;
#> coef_t = 0.999 ; coef_g = 0.640406865908954 ; rmse_mean = 1.02260178540272 ;
#> coef_t = 0.999 ; coef_g = 0.640406865908954 ; rmse_mean = 1.02260178540272 ;
#> coef_t = 0.998 ; coef_g = 0.640406865908954 ; rmse_mean = 1.0412734081074 ;
#> coef_t = 0.999 ; coef_g = 0.641406865908954 ; rmse_mean = 1.02260188447524 ;
#> coef_t = 0.999 ; coef_g = 0.639406865908954 ; rmse_mean = 1.02260188588352 ;
#> $par
#>         t         g
#> 0.9990000 0.6404069
#>
#> $value
#> [1] 1.022602
#>
#> $counts
#> function gradient
#>       18       18
#>
#> $convergence
#> [1] 0
#>
#> $message
#> [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"

## Result:
##   coef_t = 0.8688536
##   coef_g = 0.1000000
##   value = 0.9937975
```

6, Best Model

```
li_dm_1 <-
  split_a1 %>%
  get_li_dm_1(coef_t = result$par[[1]], coef_g = result$par[[2]])

mod_lm_1 <-
  li_dm_1 %>%
  {get_mod(.$train)}

li_dm_1$test %>%
  bind_cols(predict(mod_lm_1, .)) %>%
  metrics(truth = ph3, estimate = .pred) %>%
  `[[`(1, 3)
#> [1] 1.014154

li_dm_1$test %>%
  bind_cols(predict(mod_lm_1, .), "index" = as.numeric(rownames(.))) %>%
```

```r
gather(ph3, .pred, key = "type", value = "heat_load") %>%
dplyr::select(index, heat_load, type) %>%
ggplot() +
geom_line(aes(x = index, y = heat_load, color = type)) +
labs(
  title = "Obs and Linear Reg Pred of Heat Load in House 3",
  subtitle = paste0(
    "with 1-step forecasted ambient temp and ",
    "1-step forecasted radiation as input"
  ),
  x = "index", y = "heat_load (W)"
)
```