Calculating the convective velocity scale over land

Hester Bronnvik

2023-09-28

Of the methods to estimate atmospheric uplift strength and availability, w* (the Deardorff velocity or the convective velocity scale) is the only one specific to convection. In environmental studies, w* estimates the velocity of vertical air flux caused by insolation of the ground heating the air above it, which expands and rises. Thus, w* pertains specifically to thermal uplift velocity.

Calculating w* over land requires several atmospheric variables. We need to know the height of the boundary layer, the pressure, temperature, and humidity at the boundary layer height, and the pressure, temperature, surface sensible heat flux, and moisture flux at ground level. Some of these variables can be retrieved from public weather models, others must be derived from those. Here, I describe how to estimate w* in moist air over land (water vapor increases uplift potential because of its specific heat and buoyancy), which variables are needed for this, and how this can be done with real data in R.

1: How w* is calculated (the math)

The equation for w* is:

$$w^* = (\frac{g \times z \times w'\theta'_v}{\theta_v})^{1/3}$$

For which:

$$w'\theta'_v = w'T' + 0.61 \times {}^\circ C \times w'q'$$

$$w'q' = \frac{kg}{m^2} \times \frac{1}{s} \times \frac{m^3}{kg} \times \frac{kg}{kg} = \frac{\mathbf{m}}{\mathbf{s}} \times \frac{\mathbf{kg}}{\mathbf{kg}}$$

$$w'T' = H \times \frac{1}{c_p} \times \frac{1}{p}$$

$$w'T' = \frac{J}{s} \times \frac{1}{m^2} \times \frac{kg}{J} \times \frac{m^3}{kg} \times K = \frac{\mathbf{m}}{\mathbf{s}} \times \mathbf{K} = \frac{m}{s} \times {}^\circ C$$

$$w'\theta'_v = \frac{K \times m}{s} + \frac{{}^\circ C \times m}{s} \times \frac{kg}{kg} = \frac{m}{s} \times K$$

$$\theta = T(\frac{p_0}{p_1})^k = \mathbf{K}$$

$$\theta_v \approx \theta \times (1 + 0.61 \times q) = \mathbf{K}$$

We can plug these in to the original equation:

$$w^* = (\frac{g \times z \times (K \times \frac{m}{s})}{T(\frac{p_0}{p_1})^k \times (1 + 0.61 \times q)})^{1/3}$$

In units of:

$$w^* = (\frac{m \times \frac{m}{s^2} \times \frac{m}{s} \times K}{K})^{1/3}$$

$$w^* = (\frac{\frac{m^3}{s^3} \times K}{K})^{1/3}$$

$$w^* = (\frac{m^3}{s^3})^{1/3}$$

$$w^* = \frac{m}{s}$$

In these equations, the constants are:

g, the acceleration due to gravity (9.81 $\frac{m}{s^2}$).

c_p, the isobaric mass heat capacity of air at common conditions, median sea level and 296 K with humidity (1012 $\frac{J}{kg} \times \frac{1}{K}$).

p, the air density at sea level and 273 K (1.225$\frac{kg}{m^3}$).

k, the Poisson constant for dry air, the ratio of the gas constant to the specific heat (0.2854).

0.61 is a conversion factor for getting virtual potential temperature from potential temperature.

and the variables are:

z, the boundary layer height (m).

H, the surface sensible heat flux ($\frac{W}{m^2}$).

$°C$, the surface level temperature ($°C$).

p_0, the pressure at ground level (mb).

p_1, the pressure at the height of the boundary layer (mb).

T, the temperature at the height of the boundary layer (K).

q, the specific humidity at the height of the boundary layer ($\frac{kg}{kg}$).

θ, the potential temperature (K).

---

2. How to estimate w* (the code)

The variables can be derived using moisture flux, surface sensible heat flux, specific humidity at the height of the boundary layer, and pressure and temperature at both ground level and the height of the boundary layer. These variables can be retrieved from the ECMWF ERA5 reanalysis.

If we use the ECMWF Climate Data Store (CDS), the names of the necessary variables are:

geopotential

specific humidity

temperature

boundary_layer_height

surface_pressure

2m_temperature

instantaneous_moisture_flux

instantaneous_surface_sensible_heat_flux

If we use the Movebank Env-DATA system (recommended for data sets with small spatial or temporal extents), the names of the necessary variables are:

ECMWF ERA5 PL Geopotential

ECMWF ERA5 PL Specific Humidity

ECMWF ERA5 PL Temperature

ECMWF ERA5 SL Boundary Layer Height

ECMWF ERA5 SL Surface Air Pressure

ECMWF ERA5 SL Temperature (2 m above Ground)

ECMWF ERA5 SL Instantaneous Moisture Flux

ECMWF ERA5 SL Mean Surface Sensible Heat Flux

The variables at pressure level (marked PL above) can then either be interpolated to the height of the boundary layer, or, for simplicity and to reduce time (especially important for medium-to-large data sets), we can simply use the values at the highest pressure level below the boundary layer height. After this, we can plug the variables directly into the equations.

First, we ready the environment and make the functions that we need.

```
library(terra)
library(ecmwfr)
library(parallel)
library(readr)
library(tidyverse)
library(ggplot2)
library(nlcor)

# a function to take the cube root
CubeRoot<-function(x){
  sign(x)*abs(x)^(1/3)
}
# a function to calculate w*
convective <- function(data) {

  df <- data # the data input
  z <- df$blh # the boundary layer height
  phi <- df$phi # the geopotential at the layer below the boundary layer
  s_flux <- df$sflux # the surface sensible heat flux
  m_flux <- df$mflux # the moisture flux
  T_k_2m <- df$T2m # the temperature at ground level
  p0 <- df$p0 # the pressure at ground level (Pascals)
  p0 <- p0/100  # the pressure at ground level (mbar)
  q <- df$q # the humidity below the boundary layer height
  p1 <- as.numeric(df$level) # the pressure below the boundary layer height (mbar)
  T_K_blh <- df$temp # the temperature below the boundary layer height

  k <- 0.2854 # Poisson constant
  g <- 9.80665 # acceleration due to gravity
  T_c_2m <- T_k_2m - 273.15 # surface temperature in degrees Celsius
```

```
  c_p <- 1012 # the isobaric mass heat capacity of air at common conditions
  p <- 1.225 # the density of air at sea level and 273 K

  Theta_k_z <- T_K_blh*((p0/p1)^k)
  Thetav_k_z <- Theta_k_z*(1+(0.61*q))
  wT <- (s_flux * -1)/c_p/p # reverse the sign. ECMWF upward fluxes are negative
  wq <- (m_flux * -1)/p # reverse the sign

  wthetav <- wT + 0.61 * T_c_2m * wq

  w_star <- CubeRoot(phi*(wthetav/Thetav_k_z))

  return(w_star)

}
```

Then, we retrieve the necessary data from the ECMWF data store.

```
### retrieve ECMWF data for each pressure and surface level for each variable in w*
### For this many data, we need ~1 week day and night to download

# set the user ID to access the CDS API
cds.key <- # add key here
wf_set_key(user = "151388", key = cds.key, service = "cds") # add user here

# define the pressure levels at which to download data
levels <- c(seq(from = 600, to = 1000, by = 25))
levels <- levels[!levels %in% c(625, 675, 725)]

# define the variables to download
variables <- c("Geopotential", "Specific humidity", "Temperature")

#list the years with data
years <- c("2013", "2014", "2015", "2016", "2017", "2018", "2019", "2020", "2021", "2022", "2023")

# loop through each year, pressure level, and variable (462 total files, 281 GB)
lapply(years, function(z){
  lapply(levels, function(y){
    lapply(variables, function(x){
      # set up the request to the archive
      request <- list(
        # the repository
        "dataset_short_name" = "reanalysis-era5-pressure-levels",
        "product_type"   = "reanalysis",
        "variable"       = x,
        "pressure_level" = y,
        "year"           = z,
        # all months
        "month"          = c(paste0(0:12)),
        # all days
        "day"            = c(paste0(1:31)),
```

```
        # all hours
        "time"           = c("00:00", "01:00", "02:00", "03:00", "04:00", "05:00", "06:00",
                             "07:00", "08:00", "09:00", "10:00", "11:00", "12:00", "13:00",
                             "14:00", "15:00", "16:00", "17:00", "18:00", "19:00", "20:00",
                             "21:00", "22:00", "23:00"),
        # the top left and bottom right points in lat/long
        "area"           = "55/-20/0/20",
        "format"         = "netcdf",
        # the file name
        "target"         = paste0(x, "_", y, "_", z, ".nc")
      )

      file <- wf_request(user = "151388",
                         request = request,
                         transfer = TRUE,
                         path = "C:/Users/hbronnvik/Documents/storkSSFs/ecmwf_data/pressure/",
                         verbose = TRUE)
    })
  })
})

variables2 <- c("2m_temperature", "boundary_layer_height", "instantaneous_moisture_flux",
                "instantaneous_surface_sensible_heat_flux", "surface_pressure")

# loop through each year and surface level variable (55 total files, 34.4 GB)
lapply(years, function(z){
  lapply(variables2, function(x){
    request <- list(
      "dataset_short_name" = "reanalysis-era5-single-levels",
      "product_type"   = "reanalysis",
      "variable"       = x,
      "year"           = z,
      "month"          = c(paste0(0:12)),
      "day"            = c(paste0(1:31)),
      "time"           = c("00:00", "01:00", "02:00", "03:00", "04:00", "05:00", "06:00",
                           "07:00", "08:00", "09:00", "10:00", "11:00", "12:00", "13:00",
                           "14:00", "15:00", "16:00", "17:00", "18:00", "19:00", "20:00",
                           "21:00", "22:00", "23:00"),
      "area"           = "55/-20/0/20",
      "format"         = "netcdf",
      "target"         = paste0(x, "_land_", z, ".nc")
    )

    file <- wf_request(user = "151388",
                       request = request,
                       transfer = TRUE,
                       path = "C:/Users/hbronnvik/Documents/storkSSFs/ecmwf_data/single/",
                       verbose = TRUE)
  })
})
```

Next, we load in our data at every pressure level and extract the temperature, humidity, and pressure at the

level just below the boundary layer for each location.

```
# the sources for the files with data at pressure levels and at ground level
file_list_s <- list.files("/home/hbronnvik/Documents/storkSSFs/ecmwf/single/",
                          full.names = T)
file_list_p <- list.files("/home/hbronnvik/Documents/storkSSFs/ecmwf/pressure/",
                          full.names = T)

a_data <- readRDS("/home/hbronnvik/Documents/storkSSFs/burst_data/used_av_df_230612.rds")

names(a_data)[c(2,3)] <- c("lat", "long")
# find the animals migrating in July
ids <- a_data %>%
  filter(timestamp > "2023-07-01") %>%
  dplyr::select(track) %>%
  unique() %>%
  deframe()
# and remove them (no available July weather models)
a_data <- a_data %>%
  filter(!track %in% ids)
# clean up
a_data$land <- NULL
a_data$splitter <- NULL
a_data$timestamp <- as.POSIXct(a_data$timestamp, tz = "UTC")

# split the data into years to match the environmental data
data_ls <- split(a_data, year(a_data$timestamp))

# loop through each year and annotate it with its environmental data
(start_time <- Sys.time())
annotated_data <- lapply(data_ls[[11]], function(steps){
  # identify the year and hours
  d_year <- unique(year(steps$timestamp))
  stamp <- round_date(unique(steps$timestamp), unit = "hour")

  # give a notification of each year's processing
  print(paste0("Extracting environmental data for ", d_year, "."), quote = F)

  # the height of the boundary layer
  file <- grep(d_year, file_list_s[grepl("boundary_layer", file_list_s)], value = T)
  blh <- try(terra::rast(file))
  blh <- blh[[which(terra::time(blh) %in% stamp)]]

  # create a list to go through each timestamp individually
  steps_ls <- split(steps, steps$timestamp)

  # the boundary layer height at the points of interest
  # start_time <- Sys.time()
  steps <- lapply(steps_ls, function(df){
    ts <- round_date(unique(df$timestamp), unit = "hour")
    h <- blh[[which(terra::time(blh) == ts)]]
    poi_blh <- terra::extract(h, vect(df, geom = c("long", "lat")))[,2]
    df$blh <- poi_blh
```

```
  df
}) %>% reduce(rbind)
# Sys.time()-start_time

# the height of the pressure levels at the points of interest
files <- grep(d_year, file_list_p[grepl("Geopotential", file_list_p)], value = T)

# start_time <- Sys.time()
heights <- lapply(files, function(file){
  geo <- try(terra::rast(file))
  temp <- lapply(steps_ls, function(df){
    ts <- round_date(unique(df$timestamp), unit = "hour")
    geo <- geo[[which(time(geo) == ts)]]
    # extract geopotentials and convert to geopotential heights
    poi_geoH <- terra::extract(geo, vect(df, geom = c("long", "lat")))[ ,2]/9.80665
    lev <- str_split(file, "_")[[1]][2]
    h <- data.frame(poi_geoH)
    colnames(h) <- lev
    h <- h %>%
      mutate(long = df$long,
             lat = df$lat)
    return(h)
  }) %>% reduce(rbind)
}) %>%
  reduce(cbind) %>%
  # mutate(blh = poi_blh) %>%
  as.data.frame()
# Sys.time()-start_time #Time difference of 26.82361 mins

heights <- heights %>%
  dplyr::select(colnames(heights)[which(!colnames(heights) %in% c("long", "lat"))]) %>%
  mutate(long = heights[ , 2],
         lat = heights[ , 3])

steps <- full_join(steps, heights)

# take the lowest pressure level below the PBL for each point.
# in the event of the BLH being below the lowest pressure level, take the lowest pressure level.
# start_time <- Sys.time()
low_pressures <- lapply(1:nrow(steps), function(x){
  v <- steps[x, c("600","650","700","750","775","800","825","850","875",
                  "900","925","950","1000")] %>% # the columns with geopotential heights
    t() %>%
    as.data.frame() %>%
    rownames_to_column(var = "level") %>%
    rename(h = 2) %>%
    arrange(h)
  if(min(v$h) < steps$blh[x]){
    v <- v %>%
      filter(h < steps$blh[x]) %>%
      filter(h == max(h))
  }else{
    v <- v %>%
```

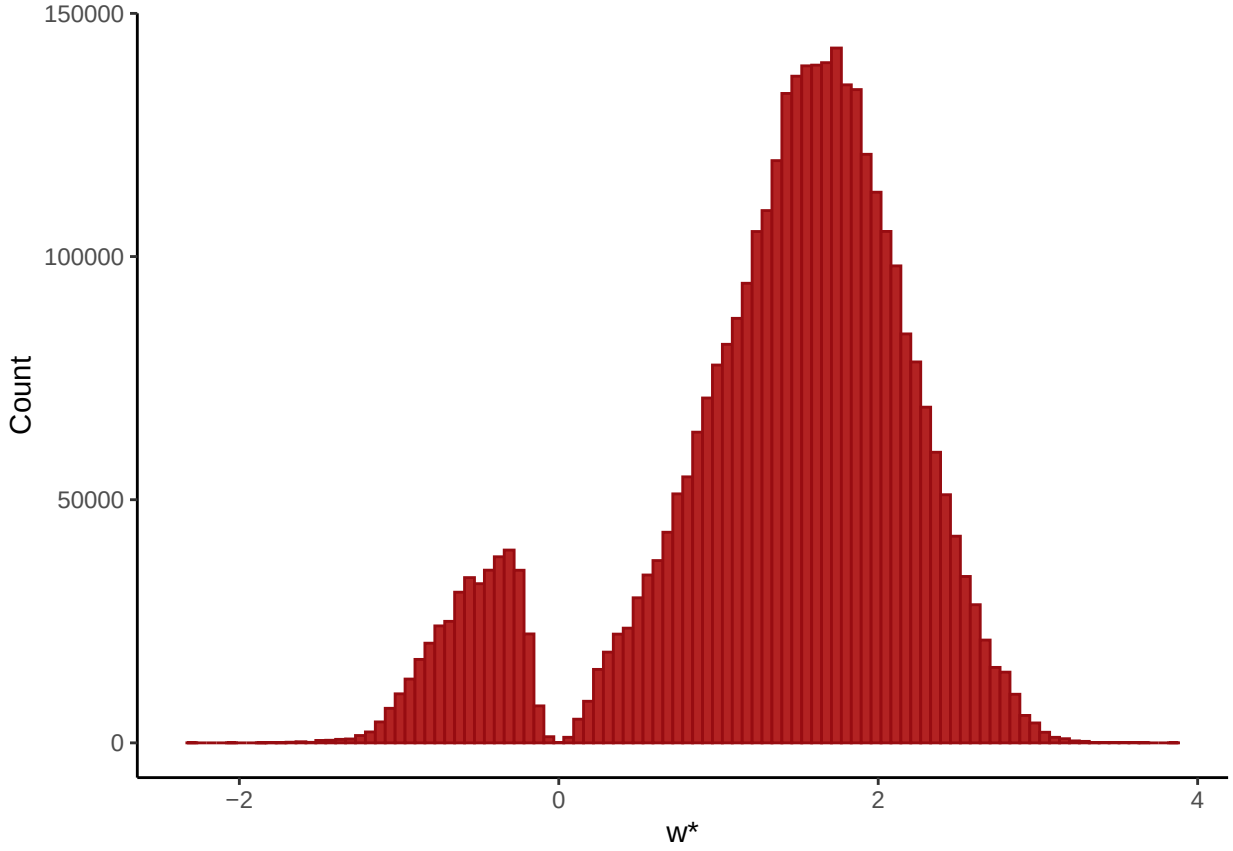
```
      filter(level != "blh") %>%
      filter(h == min(h))
  }
  v <- v %>%
    mutate(long = steps$long[x],
           lat = steps$lat[x])
}) %>% reduce(rbind)
# Sys.time()-start_time

steps <- full_join(steps, low_pressures)

# now we have the pressure levels we need to extract values from, name the files
files <- file_list_p[grep("Specific humidity|Temperature", file_list_p)]
file_levels <- paste(unique(low_pressures$level), collapse = "|")
files <- grep(paste0(file_levels), files[grepl(d_year, files)], value = T)
variables <- c("Specific humidity", "Temperature")

# the value at the level below the PBL for each pressure variable
# start_time <- Sys.time()
# split the year into different pressure levels so that only one file need be called per group
pressures <- lapply(split(steps, steps$level), function(l){
  # for each variable,
  temp <- lapply(variables, function(var){
    # call the file at the level
    file <- grep(var, grep(unique(l$level), files, value = T), value = T)
    # subset it to have only the times when the PBL was at that pressure level
    ras <- try(terra::rast(file))
    # then go through each time step individually and extract the values at each location
    val <- lapply(split(l, l$timestamp), function(lt){
      ts <- round_date(unique(lt$timestamp), unit = "hour")
      r <- ras[[which(time(ras) == ts)]]
      val <- terra::extract(r, vect(lt, geom = c("long", "lat")))[ ,2]
      val_df <- data.frame(timestamp = lt$timestamp, value = val,
                           long = lt$long, lat = lt$lat)
    }) %>% reduce(rbind)
    # tidy and save
    v <- names(ras)[1]
    variable <- ifelse(grepl("q_", v), "q", "temp")
    lev <- str_split(file, "_")[[1]][2]
    df <- val %>%
      mutate(level = lev)
    colnames(df)[which(colnames(df) == "value")] <- variable
    return(df)
  }) %>%
    reduce(cbind)
}) %>% reduce(rbind)
# Sys.time()-start_time # Time difference of 14.89281 mins

# now get the ground level data for the point of interest
files <- grep(d_year, file_list_s[!grepl("boundary", file_list_s)], value = T)
variables <- c("instantaneous_moisture_flux", "temperature",
               "instantaneous_surface_sensible", "surface_pressure")
```

```
  # start_time <- Sys.time()
  # split the year into different pressure levels so that only one file need be called per group
  singles <- lapply(variables, function(var){
    # call the file at the level
    file <- grep(var, files, value = T)
    # subset it to have only the times when the PBL was at that pressure level
    ras <- try(terra::rast(file))
    # then go through each time step individually and extract the values at each location
    val <- lapply(split(steps, steps$timestamp), function(lt){
      ts <- round_date(unique(lt$timestamp), unit = "hour")
      r <- ras[[which(time(ras) == ts)]]
      val <- terra::extract(r, vect(lt, geom = c("long", "lat")))[ ,2]
      val_df <- data.frame(timestamp = lt$timestamp, value = val, long = lt$long, lat = lt$lat)
    }) %>% reduce(rbind)
    # tidy and save
    v <- names(ras)[1]
    variable <- ifelse(grepl("t2m_", v), "T2m",
                       ifelse(grepl("ie_", v), "mflux",
                              ifelse(grepl("ishf_", v), "sflux",
                                     ifelse(grepl("sp_", v), "p0", NA))))
    colnames(val)[2] <- variable
    return(val)
  }) %>%
    reduce(cbind)
  # Sys.time()-start_time # 7.434673 mins

  singles <- singles %>%
    dplyr::select(colnames(singles)[which(!colnames(singles) %in% c("long", "lat", "timestamp"))]) %>%
    mutate(long = singles[ , 3],
           lat = singles[ , 4],
           timestamp = singles[ , 5])

  extracts <- full_join(singles, pressures[, c(1:5, 7)])
  df <- full_join(steps, extracts)
  df
}) %>%
  reduce(rbind) %>%
  as.data.frame()
Sys.time() - start_time
```

Finally, we estimate w* and explore the outcome.

```
# find the geopotential of each location at the level below the top of the boundary layer
annotated_data$phi <- lapply(1:nrow(annotated_data), function(x){
  l <- annotated_data$level[x]
  geoH <- annotated_data[x, which(colnames(annotated_data) == l)]
  phi <- geoH*9.80665
}) %>% unlist()

# calculate the w* values
annotated_data$w_star <- convective(data = annotated_data)
```

```r
# look at the distributions of w* values
ggplot(annotated_data, aes(w_star)) +
  geom_histogram(bins = 100, color = "#970C11", fill = "firebrick") +
  labs(x = "w*", y = "Count") +
  theme_classic()
```




```r
summary(annotated_data$w_star)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -2.2911  0.9784  1.5117  1.3420  1.9215  3.8555
```
