extract_huc_data() workflow

This Markdown walks through the how to:

1. run the `extract_huc_data` function
2. retrieve specific information from it's output

Load required packages and source function from R script

```
library(tools)
library(readr)
library(stringr)
library(fs)
library(dplyr)
source("extract_data_function_v2.R")
```

The function requires three inputs

- basin directory (basin_dir)
  - This is the location of the `basin_data_public_v1p2` folder. From this directory you should be able to further navigate to desired daymet mean forcing data folders (labeled 01, 02, 03, etc) via : "~/home/basin_dataset_public_v1p2/basin_mean_forcing/daymet" , and the streamflow folders should be in: "~/home/basin_dataset_public_v1p2/usgs_streamflow" . This *exact* folder structure is required for the function to work properly
- attribute directory (attr_dir)
  - location of .txt files for data attributes (camels_clim.txt, camels_geol.txt, etc)
- huc ids (huc8_names)
  - a vector of 8 digit huc 8 ids to be queried

Running function

```
##~ directories
basin_dir <- "~/CAMELS/basin_dataset_public_v1p2/"
attr_dir <- "~/CAMELS/camels_attributes_v2.0"

##~ list of hucs to query (provided as a vector)
huc8_names <- c("01013500", "08269000", "10259200")

### run function
##~ this returns a named list object with 9 items
data <- extract_huc_data(basin_dir = basin_dir,
                         attr_dir = attr_dir,
                         huc8_names = huc8_names)
```

Access output

view names of each list item

```
names(data)
```

```
## [1] "mean_forcing_daymet" "usgs_streamflow"     "camels_clim"
## [4] "camels_geol"         "camels_hydro"        "camels_name"
## [7] "camels_soil"         "camels_topo"         "camels_vege"
```

access each item

```
mean_forcing <- data$mean_forcing_daymet

### an alternative using [[]] syntax:
##~  mean_forcing <- data[["mean_forcing_daymet"]]
## OR, because this is the first item in the list:
##~  mean_forcing <- data[[1]]

##this returns a tibble/data frame containing the mean forcing data
str(mean_forcing)
```

```
## tibble [38,352 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ID           : chr [1:38352] "01013500" "01013500" "01013500" "01013500" ...
##  $ Year         : num [1:38352] 1980 1980 1980 1980 1980 1980 1980 1980 1980 1980 ...
##  $ Mnth         : chr [1:38352] "01" "01" "01" "01" ...
##  $ Day          : chr [1:38352] "01" "02" "03" "04" ...
##  $ Hr           : num [1:38352] 12 12 12 12 12 12 12 12 12 12 ...
##  $ dayl(s)      : num [1:38352] 30173 30253 30344 30408 30413 ...
##  $ prcp(mm/day): num [1:38352] 0 0 0 0 0 0 6.69 3.64 0 0 ...
##  $ srad(W/m2)   : num [1:38352] 153 145 147 146 170 ...
##  $ swe(mm)      : num [1:38352] 0 0 0 0 0 0 0 0 0 0 ...
##  $ tmax(C)      : num [1:38352] -6.54 -6.18 -9.89 -10.98 -11.29 ...
##  $ tmin(C)      : num [1:38352] -16.3 -15.2 -18.9 -19.8 -22.2 ...
##  $ vp(Pa)       : num [1:38352] 172 186 138 120 118 ...
```

```
## furthermore, we can see that each of the hucs we entered are present
unique(mean_forcing$ID)
```

```
## [1] "01013500" "08269000" "10259200"
```

visualize

```
library(ggplot2)
library(lubridate) # for dates
library(janitor) # clean column names

# clean column names
cleaned_names <- clean_names(mean_forcing)

### turn year, month, day columns into single "date" column
mean_forcing_date <- cleaned_names %>%
  ## join columns, forcing into year, month, day format
  mutate(date = ymd(paste(year, mnth, day, sep = "-")))

ggplot(mean_forcing_date, aes(date, prcp_mm_day)) +
  geom_line() +
  facet_wrap(~id)
```