An Introduction to the Reproducible Climate Futures package

Janelle Christensen

8-20-2021

About this tutorial

This vignette will walk through how to use the rcf package. Using the functions in this package will enable users to be make the choice to automatically select Global Circulation Models (GCMs) that are most representative of 4 climate futures (CFs) - Warm Wet, Warm Dry (or Damp), Hot Wet, and Hot Dry (or Damp)*.

It can also be used to manipulate the data to produce summaries of threshold values for 25 variables using one of three methods (quadrant, corner, or PCA) and duration (month, season, or year).

*GCMs will be labeled as “damp” rather than “dry” if the mean of future precipitation for climate futures labeled as “dry” in that location are greater than zero.

Expected knowledge

This package expects that you have a basic understanding of GCMs and that each one represents a plausible climate future. The GCMs in this package are based off of the downscaled climate model MACA (Multivariate Adaptive Climate Analog) Version 2. Additionally, this vignette assumes you have some knowlege of packages in the `tidyverse` specifically `readr` and the `read_csv()` function, dplyr functions such as `filter()`, `mutate()` and `select()` as well as some understanding of how to use `ggplot2`.

Learning goals of this vignette

At the end of this tutorial, you should be able to understand:

- the workflow of the rcf package
- naming conventions of .csv files that are output from the functions
- the different ways you can choose to summarize the data (month, season, year)
- a basic understanding of the different summarization methods in this package (quadrant, corner, PCA) and how to apply them

Important requirements and considerations

The files in this package, especially the raw data, are large. If you are downloading data in this package using `rcf_data()`, you will need at least *500 MB* of space. If you are using data that you have downloaded from elsewhere, specifically for the `cf_pca()` function, please pay attention to detail and ensure that anywhere you need to enter variable names that they match the column names of your dataset exactly.

Best practices in R use project directories, so this vignette will walk through the steps as if this project is using a .Rproj file inside of a directory that will hold all information and resultant data for this project. This vignette will also take use of the `here()` package, rather than relative file paths.

Package run-through

Workspace setup

First, let’s load some packages.

```
library(rcf)
library(dplyr)
library(readr)
library(here)
library(ggplot2)
library(ggrepel)
```

Then let's set the workspace that we will be using for all outputs from this package.

```
my_directory <- here::here()
```

This will set your directory to wherever the .Rproj file is located. You could alternatively create a directory in your project directory for the site you are looking at to save files there.

1. Download the data

Next, we will download our data using the `rcf_data()` function.

This function downloads all data from the `cft` package for all years from 1950 - 2099 for all 40 climate models. Because of the amount of data being downloaded, this function takes about 4 hours to run.

For this function, you can choose your units to be either "imperial" (the default) or "metric." Keep in mind that if using metric, variables that are calculated later, such as the heat index, can only be calculated with Fahrenheit. Results for those variables will convert the temperature to Fahrenheit first.

```
# raw_data <- rcf_data(SiteID = "BAND",
#                      latitude = 35.75758546,
#                      longitude = -106.3054344,
#                      directory = my_directory,
#                      units = "imperial")
```

```
raw_data <- read_csv("https://irmadev.nps.gov/DataStore/DownloadFile/660685")
#> Rows: 2191480 Columns: 10
#> -- Column specification --------------------------------------------------------
#> Delimiter: ","
#> chr  (2): gcm, units
#> dbl  (7): yr, precip, tmin, tmax, tavg, rhmin, rhmax
#> dttm (1): date
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

The output of this function is a file named "BAND.csv", which will be saved in your working directory. It includes daily values of:

Variable Name	Column Name
Minimum temperature	tmin
Maximum temperature	tmax
Precipitation	precip
Minimum relative humidity	rhmin
Maximum relative humidity	rhmax

Variable Name	Column Name
Date	date
Year	yr

If you choose to download the data from the `cft` package or another source separately, please ensure that:

- Your data includes at least gcm, date, year, minimum temperature, maximum temperature, average temperature, and precipitation - data that downloads from the cft package will include relative humidity, but the functions will not break if your dataset doesn't include relative humidity
- Data is daily observations
- The column names of those variables match the names in the table above
- Your data has a range of at least 30 years in the past and 30 years in the future

2. Calculate thresholds

Now that you have your raw data, the next step in the package workflow is to calculate the threshold values for this site.

We can calculate thresholds using the `calc_thresholds()` function.

```
thresholds <- calc_thresholds("BAND", data = raw_data, directory = my_directory, units = "imperial")
#> Adding missing grouping variables: `gcm`
#> Warning in calc_thresholds("BAND", data = raw_data, directory = my_directory, :
#> thresholds.csv generated successfully. DO NOT edit this csv in excel. File is
#> too large and data will be lost, causing errors in future calculations.
# let's look at the data

glimpse(thresholds)
#> Rows: 2,191,480
#> Columns: 35
#> Groups: gcm, yr [6,000]
#> $ gcm                      <chr> "bcc-csm1-1.rcp45", "bcc-csm1-1.rcp85", "bcc-~
#> $ date                     <dttm> 1950-01-01, 1950-01-01, 1950-01-01, 1950-01-~
#> $ yr                       <dbl> 1950, 1950, 1950, 1950, 1950, 1950, 1950, 195~
#> $ precip                   <dbl> 0.00000000, 0.00000000, 0.23214269, 0.2321426~
#> $ tmin                     <dbl> 25.184620, 25.184620, 28.711067, 28.711067, 2~
#> $ tmax                     <dbl> 43.52058, 43.52058, 46.70036, 46.70036, 40.96~
#> $ tavg                     <dbl> 34.35260, 34.35260, 37.70571, 37.70571, 32.66~
#> $ rhmin                    <dbl> 47.19806, 47.19806, 62.10468, 62.10468, 48.50~
#> $ rhmax                    <dbl> 99.00227, 99.00227, 99.97195, 99.97195, 97.07~
#> $ units                    <chr> "imperial", "imperial", "imperial", "imperial~
#> $ month                    <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
#> $ doy                      <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
#> $ halfyr                   <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
#> $ quarter                  <chr> "DJF", "DJF", "DJF", "DJF", "DJF", "DJF", "DJ~
#> $ heat_index               <dbl> 39.79095, 39.79095, 43.98931, 43.98931, 37.04~
#> $ heat_index_ec            <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FAL~
#> $ heat_index_dan           <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FAL~
#> $ temp_over_95_pctl        <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FAL~
#> $ temp_over_99_pctl        <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FAL~
#> $ temp_over_95_pctl_length <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
#> $ temp_under_freeze        <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRU~
#> $ temp_under_freeze_length <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
#> $ temp_under_5_pctl        <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FAL~
#> $ no_precip                <lgl> TRUE, TRUE, FALSE, FALSE, TRUE, TRUE, TRUE, T~
#> $ no_precip_length         <int> 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
```

```
#> $ precip_95_pctl          <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FAL~
#> $ precip_99_pctl          <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FAL~
#> $ precip_moderate         <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FAL~
#> $ precip_heavy            <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FAL~
#> $ freeze_thaw             <lgl> TRUE, TRUE, FALSE, FALSE, TRUE, TRUE, FALSE, ~
#> $ gdd                     <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, TRU~
#> $ gdd_count               <int> 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, ~
#> $ not_gdd_count           <int> 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, ~
#> $ frost                   <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, TRU~
#> $ grow_length             <dbl> 234, 234, 280, 280, 279, 279, 239, 239, 256, ~
```

As you can see, the majority of the results from this function will be `TRUE` or `FALSE` observations. This is because these are tests of whether or not a value on that day crossed over a specified threshold. `TRUE` meaning that the value is more than (or, for variables like temp_under_5_pctl, is less than) the threshold, `FALSE` meaning it is not.

The output of this function is a csv file named “BAND_thresholds.csv”

For most users of this package, there will only be one more function to run before moving on to graphing. Advanced users can continue to follow along with this vignette or choose to move onto the PCA intro vignette.

3. Calculate Climate Futures

The `cf_quadrant()` function allows users to calculate climate futures one of two ways:

1. Quadrant method - this method assigns one of 5 climate futures to each model, depending on change in precipitation and temperature from historic values, centered around one year in the future. The final values in the dataframe are the mean of all observations from all models in the quadrant, summarized by month, season or year, depending on summarization method chosen.
2. Corner method - this method assigns one of 4 climate futures to the most extreme model in each quadrant. The final values in this dataframe are the mean of all observations in that model, summarized by month, season or year, depending on the summarization method chosen.

In total, there are 6 ways you can choose to summarize the threshold data using the `cf_quadrant()` function, but here we will use the corner method and summarize by year.

```
corner_year <- cf_quadrant("BAND", data = thresholds, future_year = 2040, summarize_by = "year", method
         = "corner", directory = my_directory)
```

The results from this function are two csvs:

1. “BAND_year_summary_c.csv” - the name of this output will change based upon what you enter into the function. Here, this file is named “year_summary” as we chose to summarize by year, but if we had chosen to summarize by month or season, this would be reflected in the file name. The “c” at the end is because we chose the corner method, but would have been labeled “q” if we had chosen the quadrant method.
2. “BAND_future_means.csv” - this file will have calculated which quadrant each model falls in as well as the most extreme model in each quadrant, if we have chosen the corner method. This csv will have to be read back into R.

Let’s look at each file.

```
glimpse(corner_year)
#> Rows: 328
#> Columns: 33
#> Groups: corner, yr, time [328]
#> $ corner                  <chr> "Hot Dry", "Hot Dry", "Hot Dry", "Hot Dry", "~
#> $ yr                      <dbl> 1950, 1951, 1952, 1953, 1954, 1955, 1956, 195~
#> $ time                    <fct> Historical, Historical, Historical, Historica~
#> $ gcm                     <chr> "HadGEM2-ES365.rcp85", "HadGEM2-ES365.rcp85",~
```

```
#> $ cf                       <chr> "Hot Dry", "Hot Dry", "Hot Dry", "Hot Dry", "~
#> $ precip_yearly            <dbl> 12.71477, 18.28329, 17.81833, 16.86499, 20.20~
#> $ tmin                     <dbl> 37.02518, 36.58620, 37.27022, 36.90160, 37.91~
#> $ tmax                     <dbl> 65.81908, 64.23517, 64.39267, 64.17045, 65.18~
#> $ tavg                     <dbl> 51.42213, 50.41069, 50.83144, 50.53602, 51.54~
#> $ rhmin                    <dbl> 26.44010, 28.99078, 28.40626, 27.80363, 28.37~
#> $ rhmax                    <dbl> 74.35852, 80.46089, 76.74296, 79.30590, 76.74~
#> $ heat_index               <dbl> 62.68147, 61.22655, 61.33560, 61.11608, 62.18~
#> $ heat_index_ec            <int> 0, 0, 3, 0, 3, 0, 2, 0, 1, 1, 6, 0, 0, 0, 0, ~
#> $ heat_index_dan           <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
#> $ temp_over_95_pctl        <int> 11, 12, 19, 9, 25, 17, 28, 26, 33, 15, 22, 2,~
#> $ temp_over_99_pctl        <int> 0, 0, 7, 0, 3, 2, 4, 0, 7, 1, 9, 0, 0, 3, 0, ~
#> $ temp_over_95_pctl_length <int> 3, 3, 11, 8, 6, 6, 11, 7, 13, 7, 18, 1, 7, 11~
#> $ temp_under_freeze        <int> 135, 134, 144, 153, 133, 155, 159, 152, 148, ~
#> $ temp_under_freeze_length <int> 58, 52, 42, 59, 44, 60, 61, 44, 35, 46, 34, 3~
#> $ temp_under_5_pctl        <int> 45, 31, 14, 15, 17, 24, 20, 24, 27, 8, 6, 12,~
#> $ no_precip                <int> 293, 271, 264, 273, 252, 287, 293, 300, 271, ~
#> $ no_precip_length         <int> 36, 29, 21, 26, 19, 28, 38, 41, 23, 29, 33, 2~
#> $ precip_95_pctl           <int> 3, 5, 5, 5, 3, 3, 2, 1, 10, 5, 5, 3, 8, 5, 6,~
#> $ precip_99_pctl           <int> 0, 0, 2, 0, 1, 1, 0, 0, 0, 1, 1, 0, 4, 2, 1, ~
#> $ precip_moderate          <int> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 3, 0, 0, ~
#> $ precip_heavy             <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
#> $ freeze_thaw              <int> 92, 108, 98, 120, 88, 123, 113, 117, 115, 124~
#> $ gdd                      <int> 263, 252, 251, 242, 266, 242, 253, 245, 234, ~
#> $ gdd_count                <int> 203, 209, 179, 194, 220, 203, 177, 218, 190, ~
#> $ not_gdd_count            <int> 44, 49, 42, 58, 42, 59, 33, 52, 36, 34, 32, 4~
#> $ frost                    <int> 35, 23, 31, 31, 37, 34, 47, 33, 18, 29, 48, 2~
#> $ grow_length              <dbl> 268, 273, 262, 226, 278, 245, 256, 237, 212, ~
#> $ units                    <chr> "imperial", "imperial", "imperial", "imperial~
```

Because there are 2 csvs that come out of this function, the csv for the future means must be read in manually.

```
means <- read_csv(here::here("BAND_future_means.csv"))
#> Rows: 40 Columns: 7
#> -- Column specification --------------------------------------------------------
#> Delimiter: ","
#> chr (3): gcm, cf, corner
#> dbl (4): precip_change, tmax_change, tmin_change, tavg_change
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message.
glimpse(means)
#> Rows: 40
#> Columns: 7
#> $ gcm           <chr> "bcc-csm1-1-m.rcp45", "bcc-csm1-1-m.rcp85", "bcc-csm1-1.~
#> $ precip_change <dbl> -0.05280980, -0.00761095, -0.62885974, 0.60603489, 0.331~
#> $ tmax_change   <dbl> 3.279025, 4.144630, 3.680522, 4.412027, 4.532049, 5.0138~
#> $ tmin_change   <dbl> 2.998992, 3.886409, 2.884021, 3.776245, 3.524543, 3.8334~
#> $ tavg_change   <dbl> 3.139008, 4.015520, 3.282272, 4.094136, 4.028296, 4.4236~
#> $ cf            <chr> "Warm Dry", "Central", "Warm Dry", "Hot Wet", "Central",~
#> $ corner        <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
```

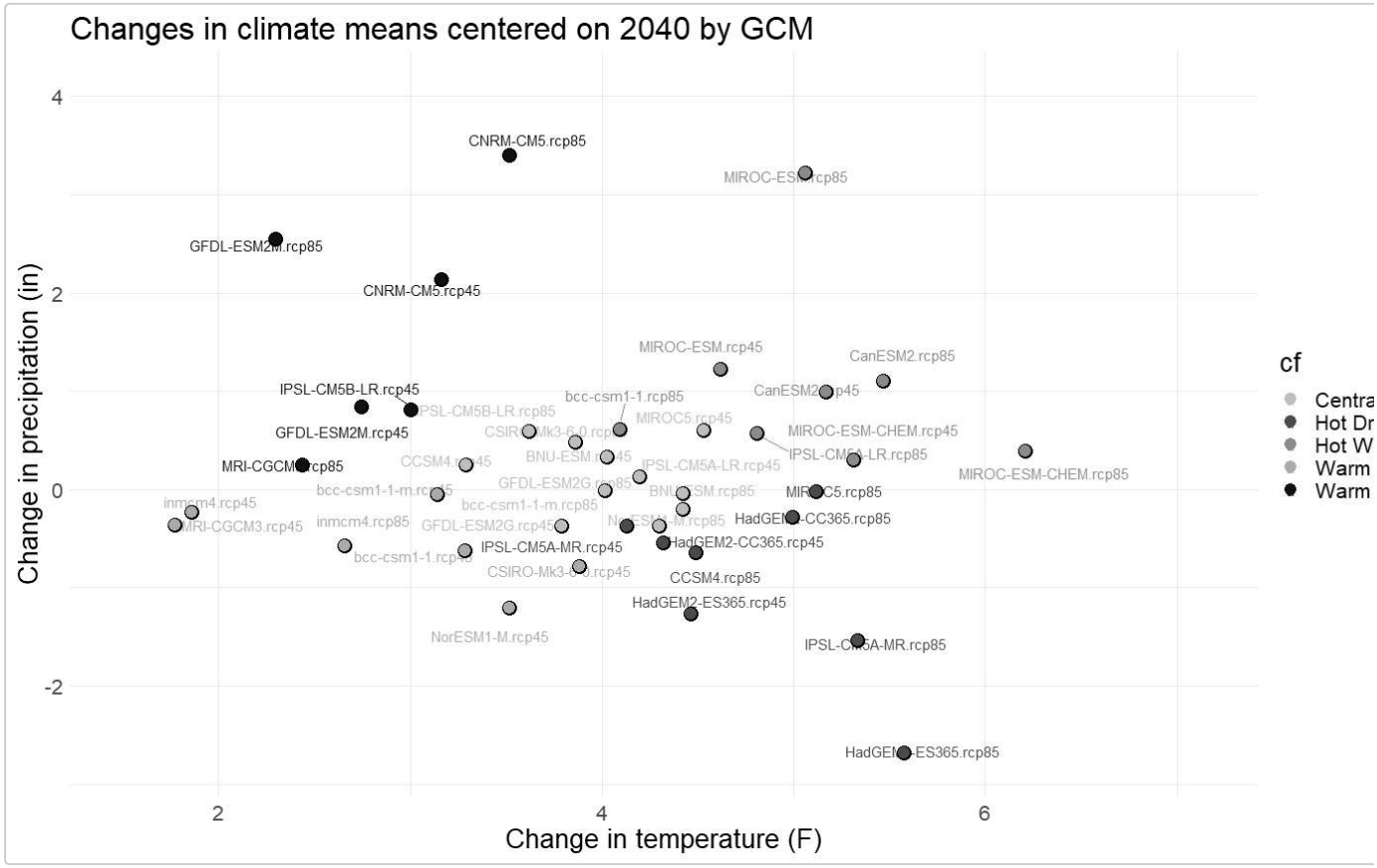
With these two files, we can now move on to making graphs.

4. Graphing

Let's start with two graphs that will show us where the models fall in relation to each other based on changes in temperature and precipitation, centered around our chosen year of 2040.

First, a graph that shows all models by color in their respective quadrants.

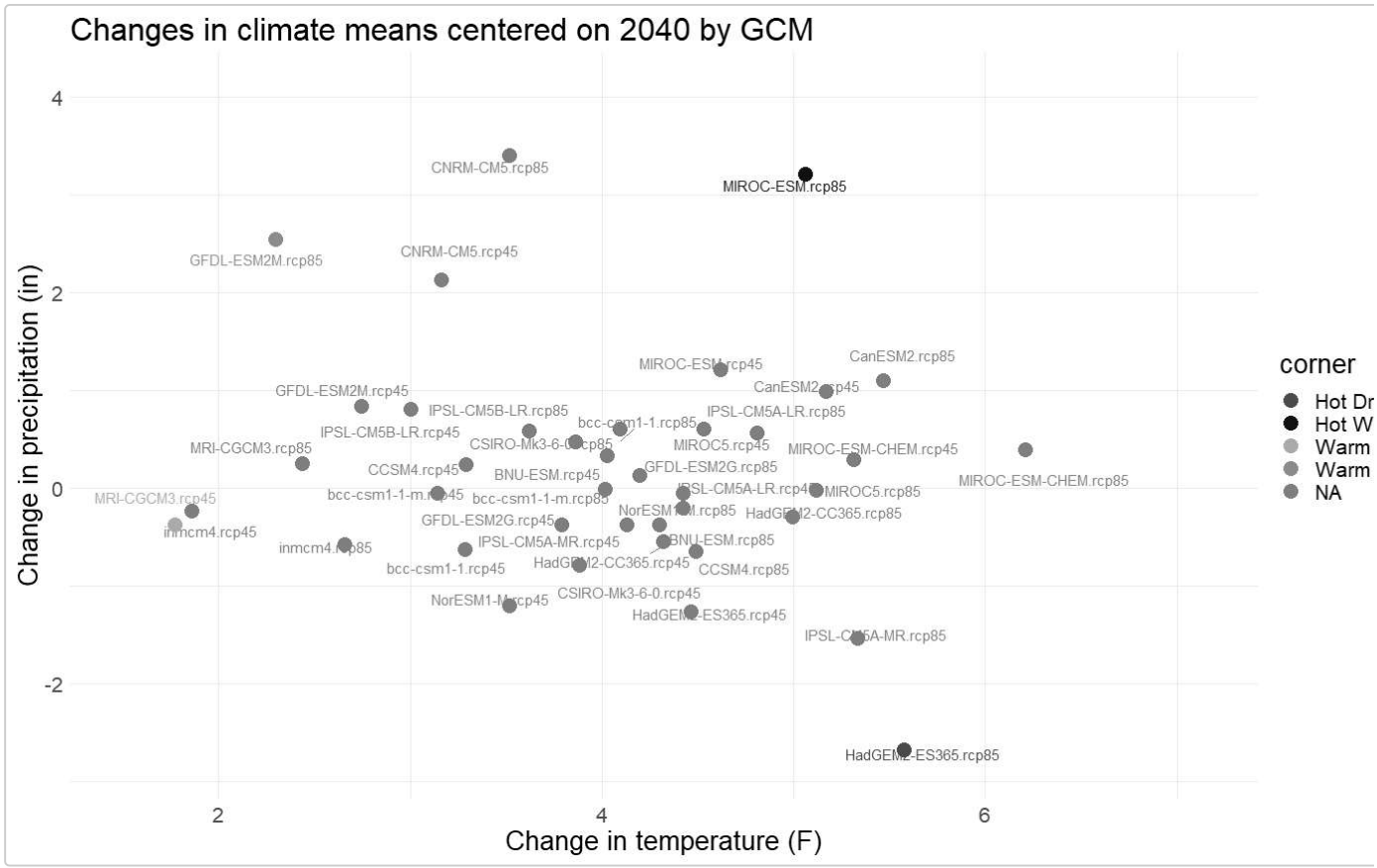
```
ggplot(means, aes(x = tavg_change,
                  y = precip_change,
                  xmin = min(tavg_change) - 0.15 * min(tavg_change),
                  xmax = max(tavg_change) + 0.15 * max(tavg_change),
                  ymin = min(precip_change) - 0.15 * min(precip_change),
                  ymax = max(precip_change) + 0.15 * max(precip_change))) +
  geom_text_repel(aes(label = gcm,
                      color = cf),
                  position = position_jitter(0,.2)) +
  geom_point(size=5,colour="black") +
  geom_point(aes(color = cf),
             size = 4) +
  labs(title = "Changes in climate means centered on 2040 by GCM",
       x = "Change in temperature (F)",
       y = "Change in precipitation (in)") +
  scale_color_manual(values = c("gray", "#E10720",  "#8386CC", "darksalmon", "#12045C")) +
  scale_fill_manual(values = c("gray", "#E10720",  "#8386CC", "#darksalmon", "#12045C")) +
  theme_minimal() +
  theme(text = element_text(size = 20))
```



Second, because we have chosen the corner method, let's make a graph that highlights the selected models.

```
ggplot(means, aes(x = tavg_change,
                  y = precip_change,
                  xmin = min(tavg_change) - 0.15 * min(tavg_change),
                  xmax = max(tavg_change) + 0.15 * max(tavg_change),
```

```
                ymin = min(precip_change) - 0.15 * min(precip_change),
                ymax = max(precip_change) + 0.15 * max(precip_change))) +
  geom_text_repel(aes(label = gcm,
                      color = corner),
                  position = position_jitter(0,.2)) +
  geom_point(aes(label = gcm,
                 color = corner),
             size = 5) +
  labs(title = "Changes in climate means centered on 2040 by GCM",
       x = "Change in temperature (F)",
       y = "Change in precipitation (in)") +
  scale_color_manual(values = c("#E10720",  "#12045C", "darksalmon", "#8386CC")) +
  scale_fill_manual(values = c("#E10720",  "#12045C", "darksalmon", "#8386CC")) +
  theme_minimal() +
  theme(text = element_text(size = 20))
#> Warning: Ignoring unknown aesthetics: label
```



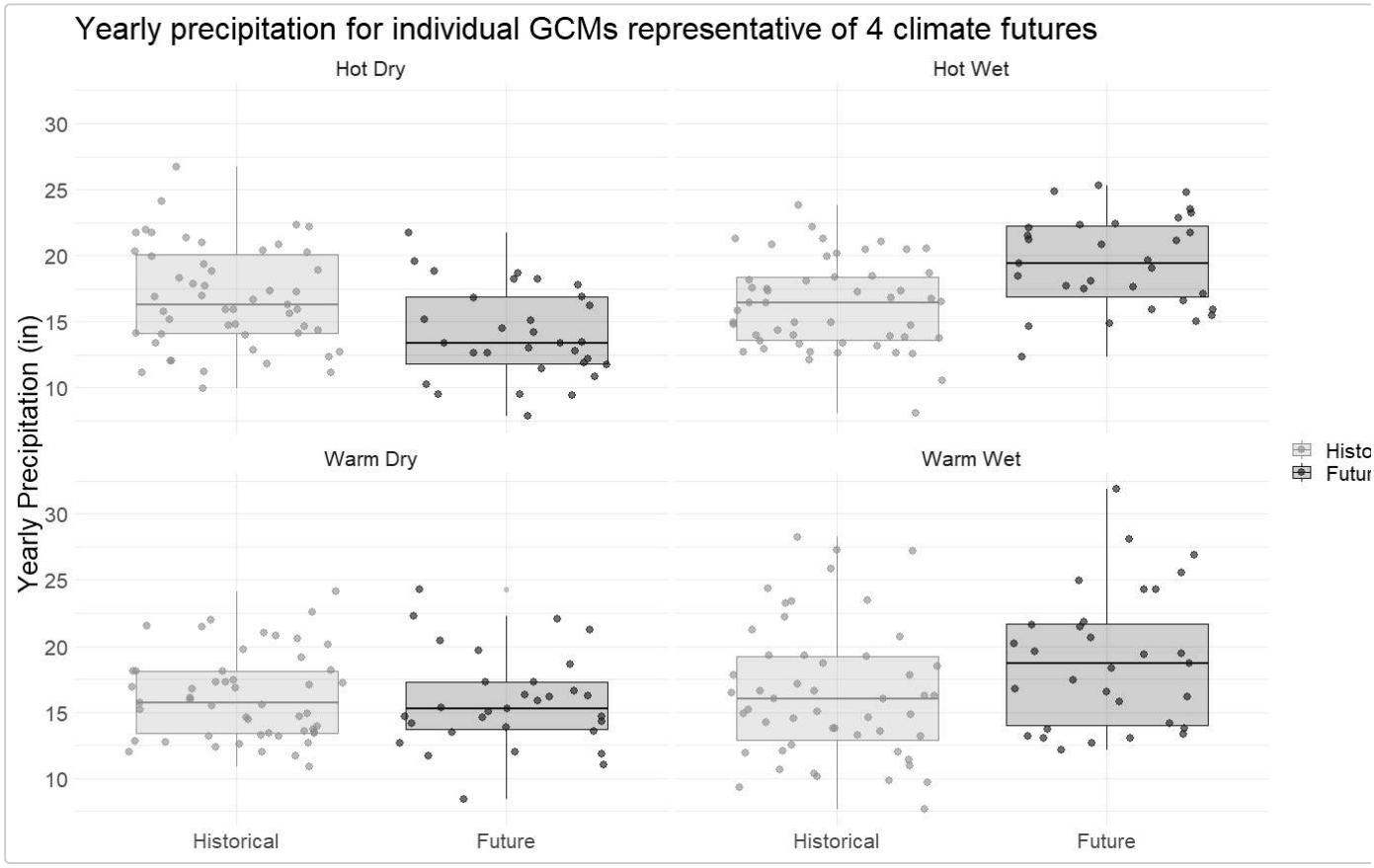
Now that we have seen where our models lie in relation to each other, let's make some plots using the threshold data.

How does the precipitation change over time in the 4 climate futures?

```
# corner_year$time <- factor(corner_year$time, levels = c("Historical", "Future"))

ggplot(data = corner_year, aes(x = time, y = precip_yearly)) +
  geom_boxplot(aes(color = time,
                   fill = time),
               alpha = 0.2) +
  geom_jitter(aes(color = time),
              size = 2.5,
              alpha =.6) +
```

```
  facet_wrap(~corner) + #facet_grid for all next to each other
  scale_color_manual(values = c("#8386CC", "#12045C")) +
  scale_fill_manual(values = c("#8386CC", "#12045C")) +
  labs(y = "Yearly Precipitation (in)",
       title = "Yearly precipitation for individual GCMs representative of 4 climate futures") +
  theme_minimal() +
  theme(text = element_text(size = 20),
        legend.title = element_blank(),
        axis.title.x = element_blank())
```

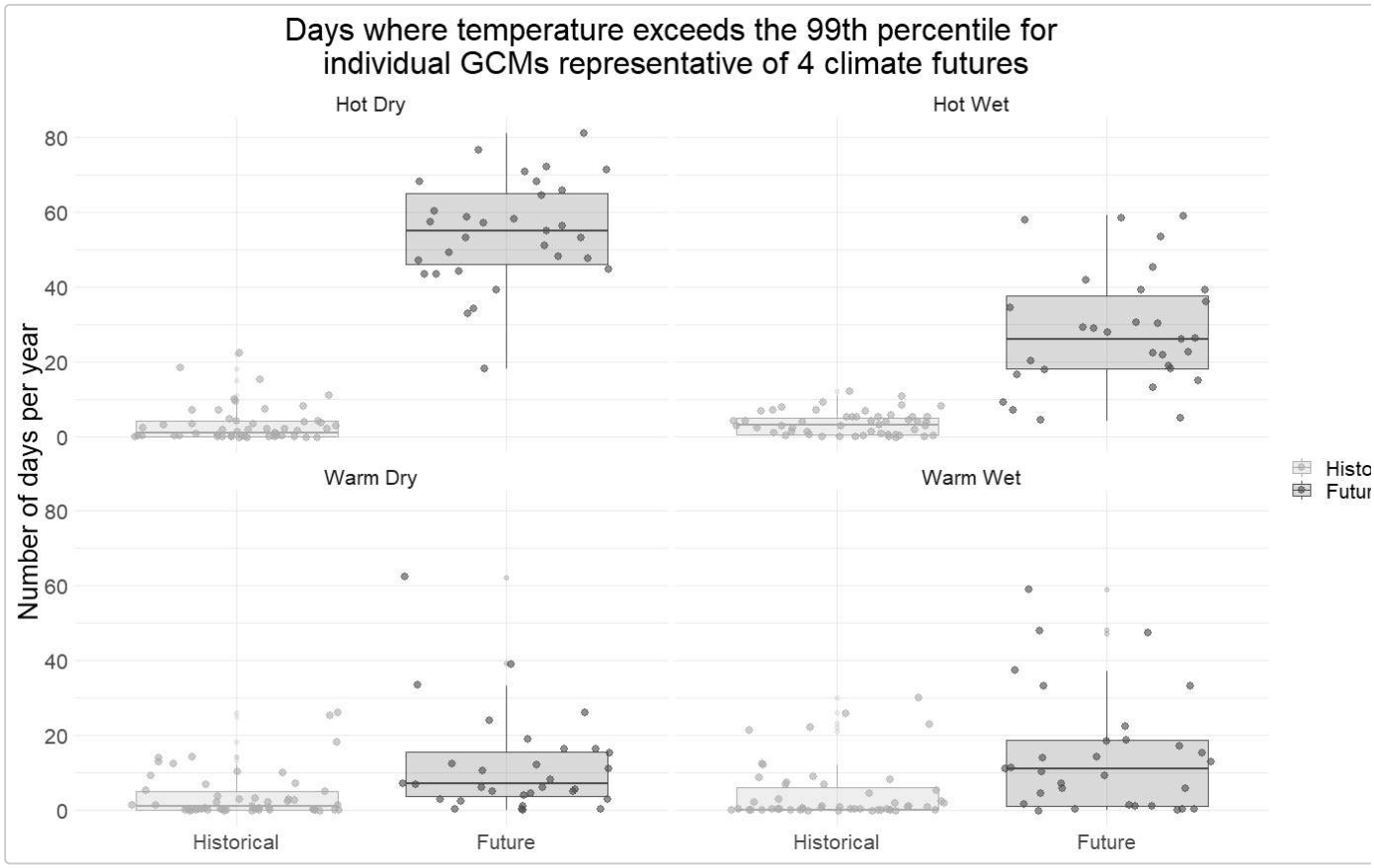


This graph could be made for any of the variables in the output from the `cf_quadrant()` function, so let's look at a few more.

How do number of days that exceed the historical 99th percentile of heat compare in the past and future?

```
ggplot(data = corner_year, aes(x = time, y = temp_over_99_pctl)) +
  geom_boxplot(aes(color = time,
                   fill = time),
               alpha = 0.2) +
  geom_jitter(aes(color = time),
              size = 2.5,
              alpha =.6) +
  facet_wrap(~corner) +
  scale_color_manual(values = c("darksalmon", "#E10720")) +
  scale_fill_manual(values = c("darksalmon", "#E10720")) +
  labs(y = "Number of days per year",
       title = "Days where temperature exceeds the 99th percentile for\n individual GCMs representative
         of 4 climate futures") +
  theme_minimal() +
  theme(text = element_text(size = 20),
        legend.title = element_blank(),
```

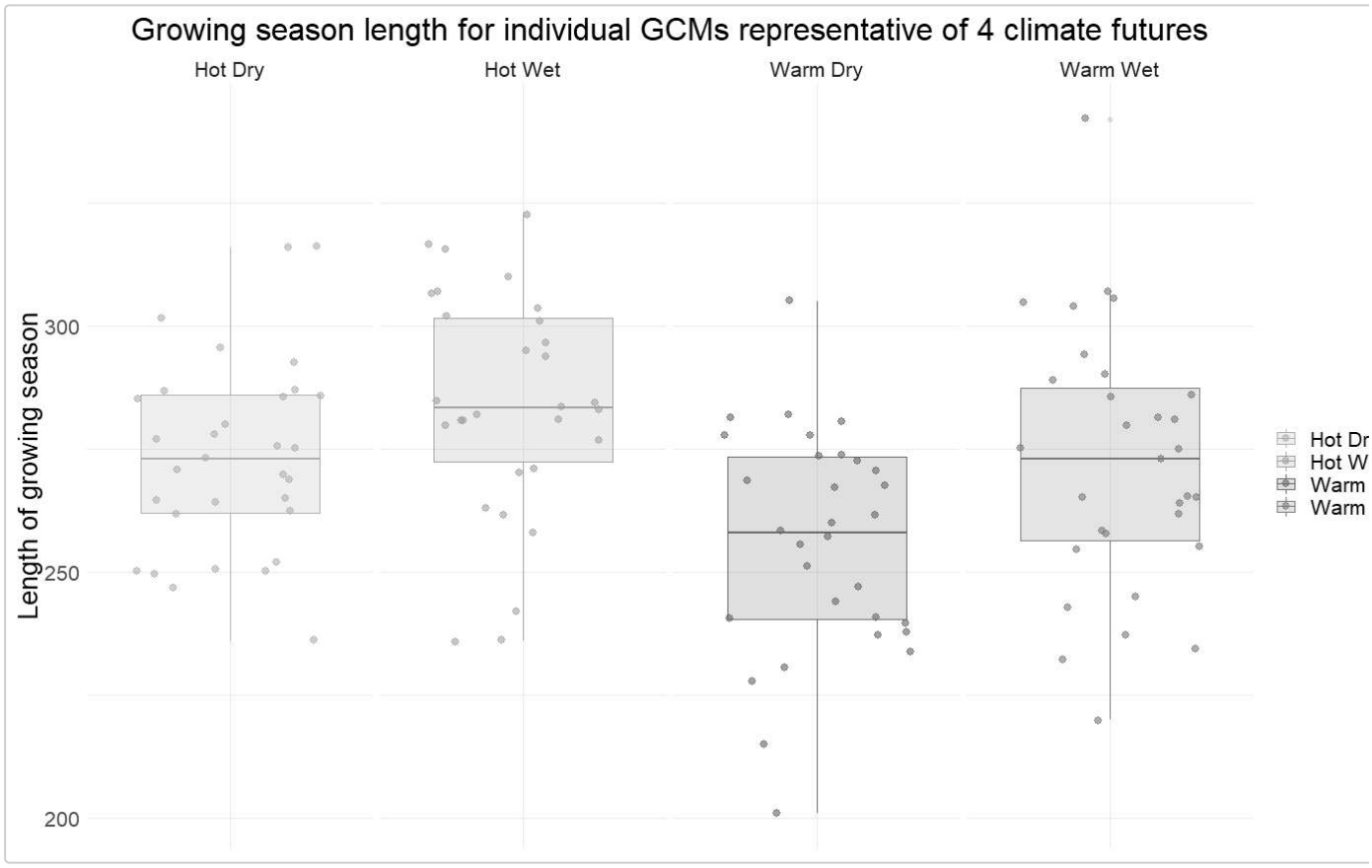
```
        axis.title.x = element_blank(),
        plot.title = element_text(hjust = 0.5))
```



What about the growing season length between models? How does that compare between the four climate futures?

```
corner_year_future <- corner_year %>%
filter(time %in% c("Future"))

ggplot(data = corner_year_future, aes(x = time, y = grow_length)) +
  geom_boxplot(aes(color = corner,
                   fill = corner),
               alpha = 0.2) +
  geom_jitter(aes(color = corner),
              size = 2.5,
              alpha =.6) +
  facet_grid(~corner) +
  scale_color_manual(values = c("#8FD834", "#72CC50", "#019875", "#00AEAD")) +
  scale_fill_manual(values = c("#8FD834", "#72CC50", "#019875", "#00AEAD")) +
  labs(y = "Length of growing season",
       title = "Growing season length for individual GCMs representative of 4 climate futures") +
  theme_minimal() +
  theme(text = element_text(size = 20),
        legend.title = element_blank(),
        axis.title.x = element_blank(),
        plot.title = element_text(hjust = 0.5),
        axis.text.x = element_blank())
```



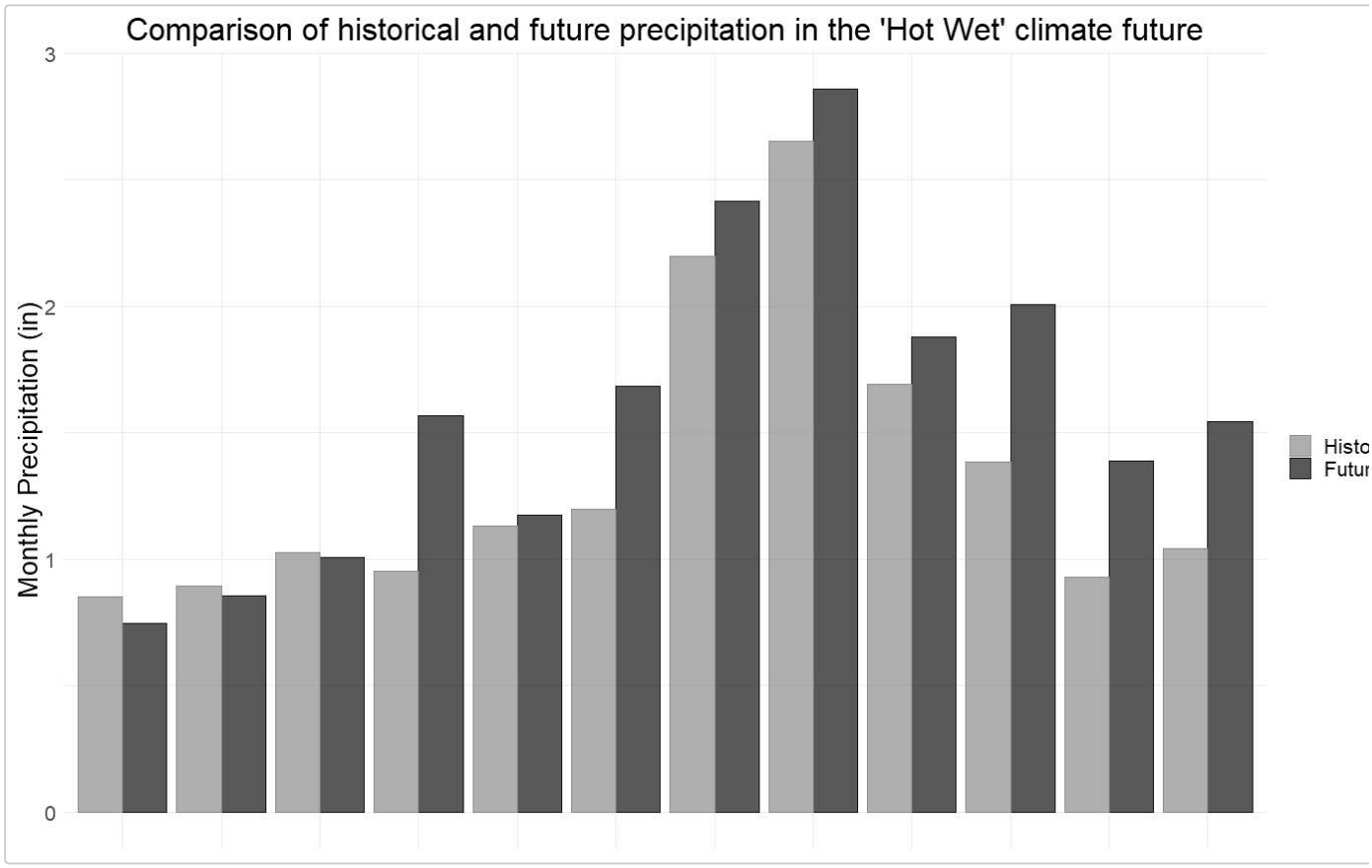
What if we want to compare by season or month? We can use the `cf_quadrant()` function again, this time summarizing by either month or season.

```
corner_month <- cf_quadrant("BAND", data = thresholds, future_year = 2040, summarize_by = "month",
         method = "corner", directory = my_directory)
```

We can then graph by month or season by climate future, or across all climate futures.

```
corner_month_hw <- corner_month %>%
  filter(corner %in% c("Hot Wet")) %>%
  mutate(month = lubridate::month(month, label = TRUE))

ggplot(data = corner_month_hw, aes(x = month, y = precip_monthly)) +
  geom_col(aes(fill = time,
               color = time),
           position = "dodge",
           alpha = 0.7) +
  scale_color_manual(values = c("#8386CC", "#12045C")) +
  scale_fill_manual(values = c("#8386CC", "#12045C")) +
  theme_minimal() +
  theme(text = element_text(size = 20),
        legend.title = element_blank(),
        axis.title.x = element_blank(),
        plot.title = element_text(hjust = 0.5),
        axis.text.x = element_blank()) +
  labs(y = "Monthly Precipitation (in)",
       title = "Comparison of historical and future precipitation in the 'Hot Wet' climate future")
```



5. Advanced Users

For users that want to try a more advanced approach to finding climate futures, the Introduction to using PCA for model selection(INSERT LINK) tutorial will walk you through the process.