Chennai Water Analysis

Ibrahim Uruc Tarim

02 June, 2024

The Chennai Water Data

It's March of 2019 in Chennai, India, and as an analyst working for the city of Chennai you must begin your quarterly report on the city's water supply and demand. Since Chennai is the biggest cultural, economic and educational center of south India, your report will impact 10 million people by informing policy and providing insight into the management of water resources in the city.

Chennai lies on the coast of the Bay of Bengal, but must rely on four nearby fresh water reservoirs that hold a combined 11,057 million cubic feet (mcft) of water. In the middle of 2019, officials announced that they had run out of water in these reservoirs. Many of the people in Chennai resorted to hand-pumped wells, while others were able to get water from tanker trucks that carried in 10 million liters a day. Now, thanks to a couple of healthy monsoon seasons, the city's water supply has been mostly restored. You can read more about the crisis in an NPR article here Chennai Water Crisis

As for the data, we provide you with two Kaggle datasets that contain water data for each of four reservoirs every day since 2004. "clevels" contains data about the water levels (millions of cubic feet, $mcft$) for each reservoir. "crainfall" has data on how much rainfall (in mm) each area recieved that day. To do well on this lab you will need to think critically about these variables to extract some information that may not be explicit in the data, but may useful in your report to city officials.

Cleaning and EDA

Our first step is to:

- load in the data,
- clean it
- perform some exploratory analysis.
- Make a rough draft plot of just a couple of the trends you want to explore.

Things to do in this chunk:

- Rename variables
- Ensure all variables have the correct class
- Perform some EDA
- We'll tackle month, year, and date extraction in the next section
- Use these dataframes separately for now

```
library(dplyr)
#load in data
#relative path from assignments folder
clevels <- read_csv("data/chennai_reservoir_levels.csv")
```

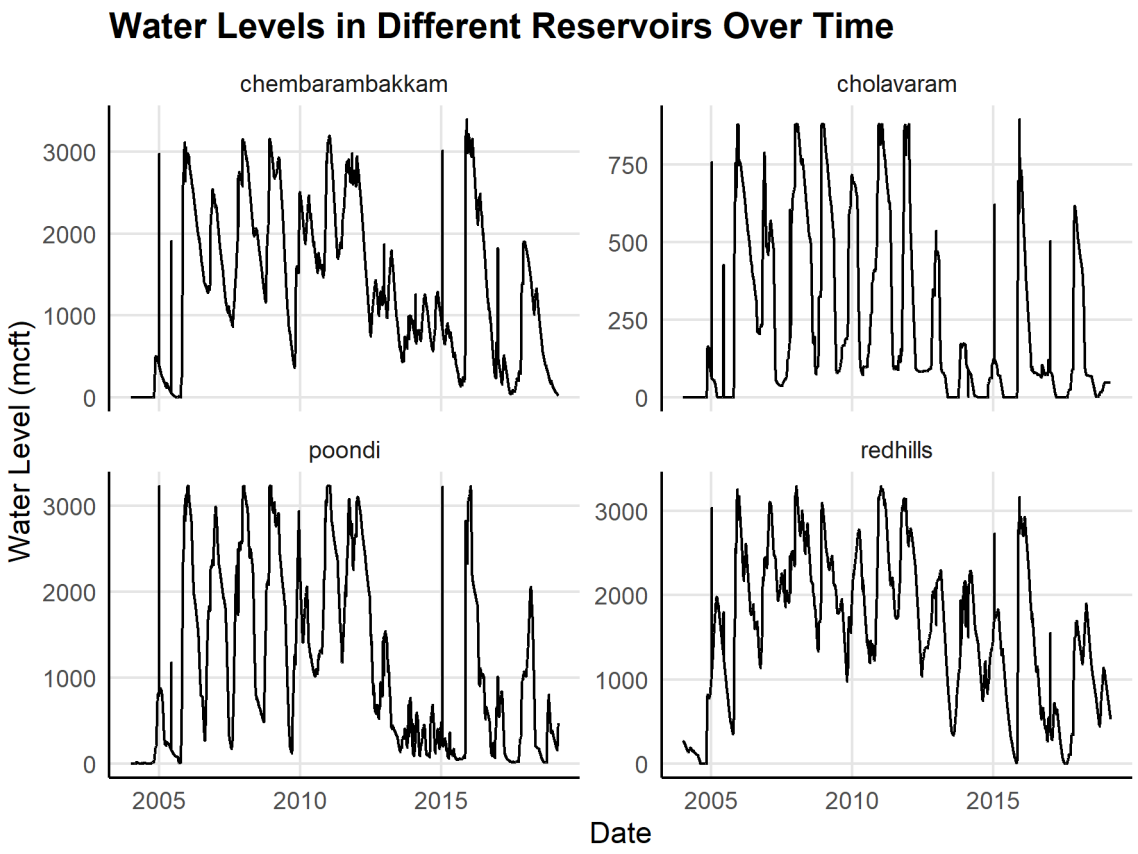
```
## Rows: 5538 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
```

```
## dbl  (4): poondi, cholavaram, redhills, chembarambakkam
## date (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.
```

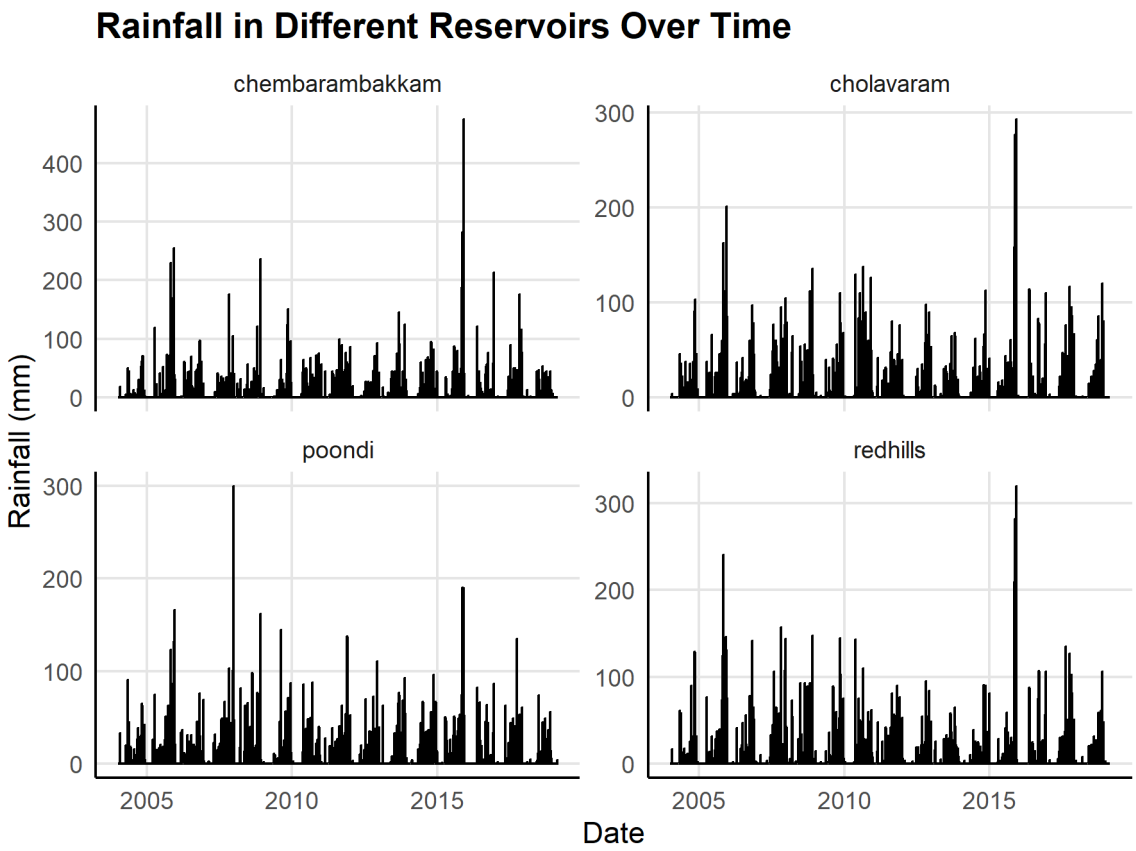
```
crain <- read_csv("data/chennai_reservoir_rainfall.csv")
```

```
## Rows: 5538 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (4): poondi, cholavaram, redhills, chembarambakkam
## date (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 str() function already confirmed that the variables are of the correct class.
#I did not need to change the names of variables, both datasets has the same names.

clevels_long <- clevels %>%
  pivot_longer(
    cols = -date,
    names_to = "reservoir",
    values_to = "level"
  )

crain_long <- crain %>%
  pivot_longer(
    cols = -date,
    names_to = "reservoir",
    values_to = "rainfall"
  )

#I converted the clevels and crain datasets to long format to facilitate easier subsetting, analysis, a

#explore

#For water levels:
ggplot(clevels_long, aes(x = date, y = level)) +
  geom_line() +
  facet_wrap(~ reservoir, scales = "free_y") +
  ggtitle("Water Levels in Different Reservoirs Over Time") +
  xlab("Date") +
  ylab("Water Level (mcft)") +
  first_custom()
```




```
#For rainfall:
ggplot(crain_long, aes(x = date, y = rainfall)) +
  geom_line() +
  facet_wrap(~ reservoir, scales = "free_y") +
  ggtitle("Rainfall in Different Reservoirs Over Time") +
  xlab("Date") +
  ylab("Rainfall (mm)") +
  first_custom()
```



Dates

I've found the lubridate package pretty useful for working with dates, because it can be a challenge to get dates to group the correct way. Date objects have some support in ggplot and other packages, but they rarely work in any other format than "YYYY-mm-dd" ", so its difficult to only reference a month or year directly and maintain the integrity of the date object. Lubridate has a couple of intuitive functions we can use on the fly to easily extract month, day, and year. We do lose the ability to operate with them as date objects, but we can recover some of that information with labeling in ggplot. Here, perform some more EDA while grouping by year, month, and year day.

Things to do: - EDA with extracting data in three ways: - year (2004:2019) **year()** - month (1:12) **month()** - year day (1:365/366) **yday()**

```
library(patchwork)
library(RColorBrewer)
library(scales)
```

```
##
## Attaching package: 'scales'
```

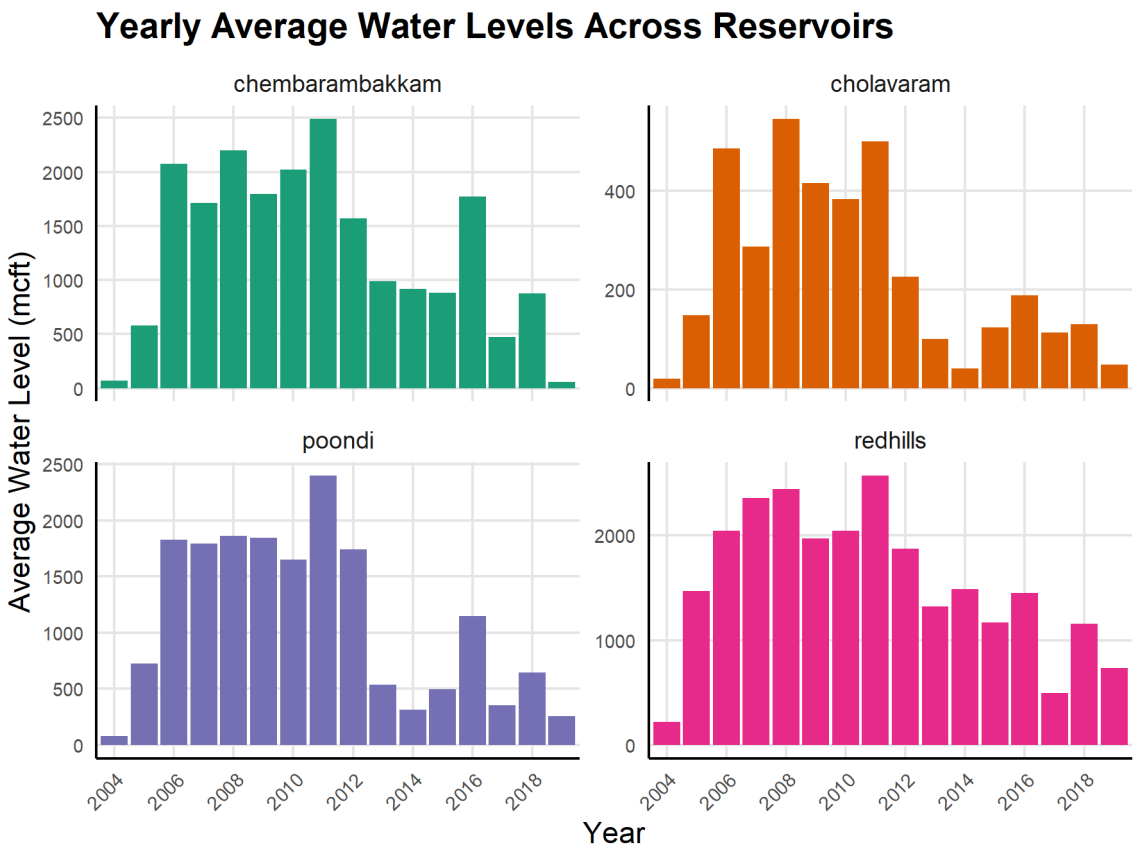
```
## The following object is masked from 'package:purrr':
##
##     discard
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```r
clevels_long$year <- year(clevels_long$date)
clevels_long$month <- month(clevels_long$date)
clevels_long$yday <- yday(clevels_long$date)
crain_long$year <- year(crain_long$date)
crain_long$month <- month(crain_long$date)
crain_long$yday <- yday(crain_long$date)


# Yearly Average Water Level Across Reservoirs
clevels_long_avg <- clevels_long %>%
  group_by(year, reservoir) %>%
  summarize(avg_level = mean(level, na.rm = TRUE)) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'year'. You can override using the
## `.groups` argument.
```

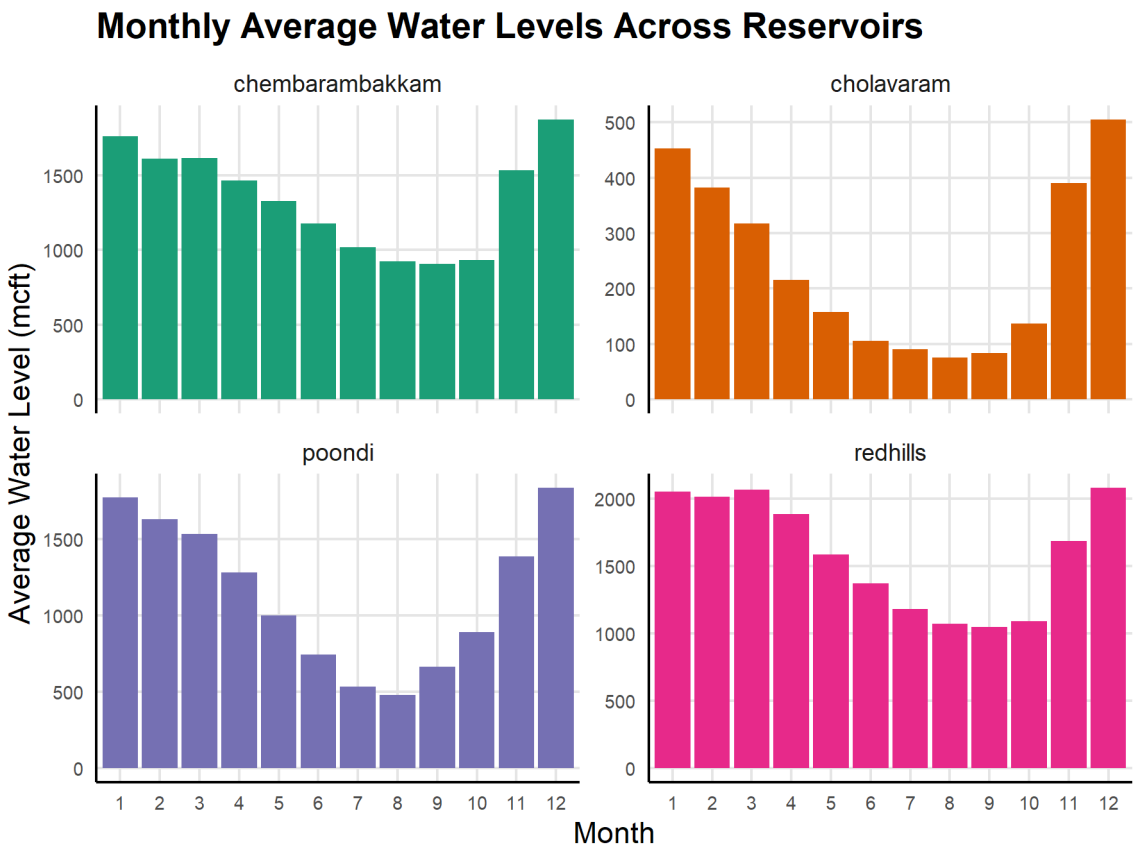
```r
ggplot(clevels_long_avg, aes(x = factor(year), y = avg_level, fill = reservoir)) +
  geom_col(aes(x = factor(year), y = avg_level)) +
  facet_wrap(~ reservoir, scales = "free_y") +
  ggtitle("Yearly Average Water Levels Across Reservoirs") +
  xlab("Year") +
  ylab("Average Water Level (mcft)") +
  first_custom() +
  scale_x_discrete(breaks = seq(2004, 2019, by = 2)) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 7),
        axis.text.y = element_text(size = 7),
        legend.position = "none") +
  scale_fill_brewer(palette = "Dark2")
```




```
# Monthly Average Water Level Across Reservoirs
clevels_monthly_avg <- clevels_long %>%
  group_by(month, reservoir) %>%
  summarize(avg_level = mean(level, na.rm = TRUE)) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'month'. You can override using the
## `.groups` argument.
```

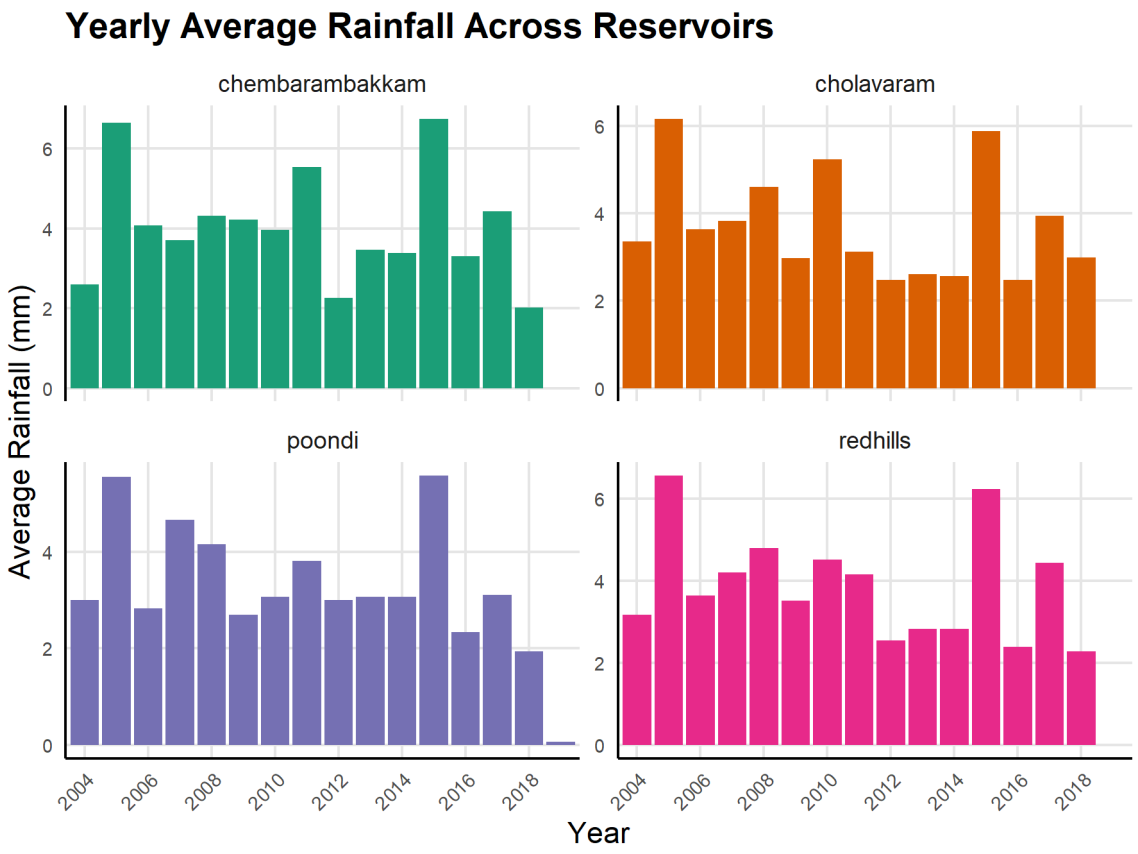
```
ggplot(clevels_monthly_avg, aes(x = factor(month), y = avg_level, fill = reservoir)) +
  geom_col() +
  facet_wrap(~ reservoir, scales = "free_y") +
  ggtitle("Monthly Average Water Levels Across Reservoirs") +
  xlab("Month") +
  ylab("Average Water Level (mcft)") +
  first_custom() +
  theme(axis.text.x = element_text(size = 7),
        axis.text.y = element_text(size = 7),
        legend.position = "none") +
  scale_fill_brewer(palette = "Dark2")
```




```
# Yearly Average Rainfall Across Reservoirs

crain_yearly_avg <- crain_long %>%
  group_by(year, reservoir) %>%
  summarize(avg_rainfall = mean(rainfall, na.rm = TRUE)) %>%
  ungroup()
```

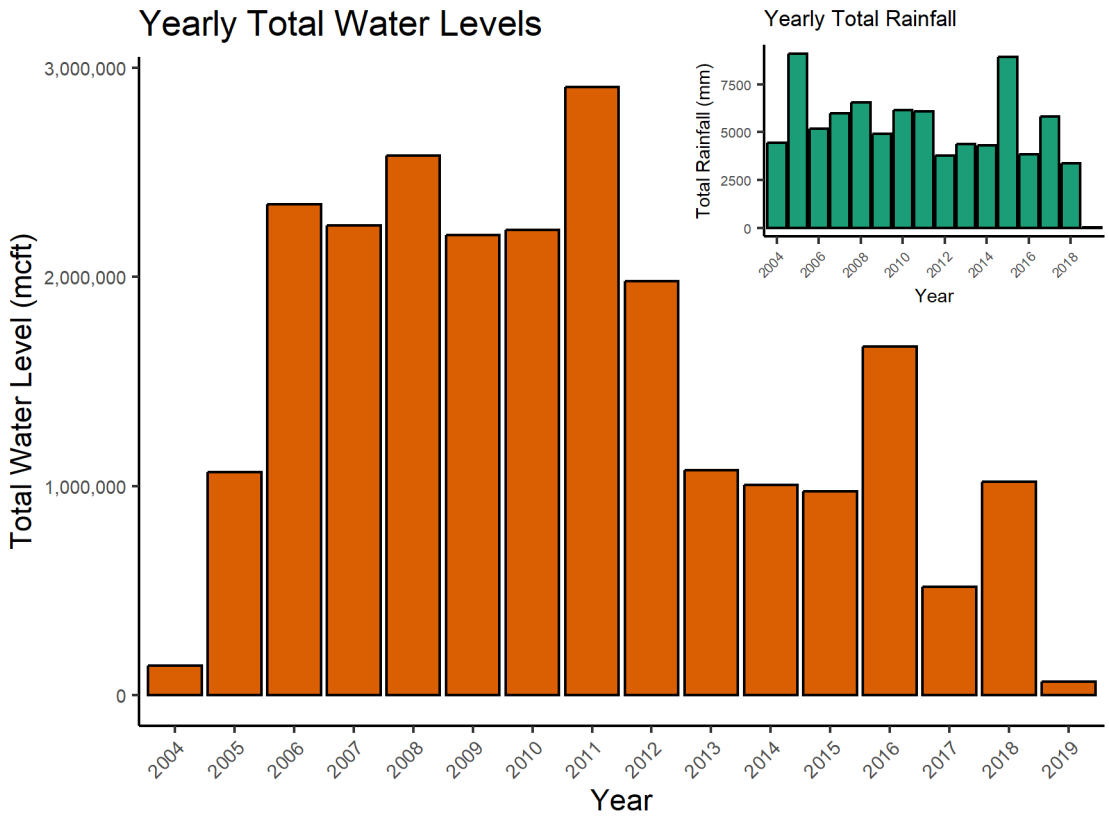
```
## `summarise()` has grouped output by 'year'. You can override using the
## `.groups` argument.
```

```
ggplot(crain_yearly_avg, aes(x = factor(year), y = avg_rainfall, fill = reservoir)) +
  geom_col() +
  facet_wrap(~ reservoir, scales = "free_y") +
  ggtitle("Yearly Average Rainfall Across Reservoirs") +
  xlab("Year") +
  ylab("Average Rainfall (mm)") +
  first_custom() +
  scale_x_discrete(breaks = seq(2004, 2019, by = 2)) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 7),
        axis.text.y = element_text(size = 7),
        legend.position = "none") +
  scale_fill_brewer(palette = "Dark2")
```




```r
# Yearly Total Water Level and Rainfall Across All Reservoirs
yearly_sum <- clevels_long %>%
  group_by(year) %>%
  summarise(total_level = sum(level))

yearly_rainfall_sum <- crain_long %>%
  group_by(year) %>%
  summarise(total_rainfall = sum(rainfall))

leveltotal <- ggplot(yearly_sum, aes(x = factor(year), y = total_level)) +
  geom_col(fill = "#D95F02", col = "black") +
  ggtitle("Yearly Total Water Levels") +
  xlab("Year") +
  ylab("Total Water Level (mcft)") +
  theme_classic() +
  scale_x_discrete(breaks = seq(2004, 2019, by = 1)) +
  scale_y_continuous(labels = comma) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 7),
        axis.text.y = element_text(size = 7))

rainfalltotal <- ggplot(yearly_rainfall_sum, aes(x = factor(year), y = total_rainfall)) +
  geom_col(fill = "#1B9E77", col = "black") +
  ggtitle("Yearly Total Rainfall") +
  xlab("Year") +
  ylab("Total Rainfall (mm)") +
  theme_classic() +
```

```
  scale_x_discrete(breaks = seq(2004, 2019, by = 2)) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 5),
        axis.text.y = element_text(size = 5),
        title = element_text(size = 7),
        axis.title = element_text(size =7))

totalvis <- leveltotal + inset_element(rainfalltotal, left = 0.61, bottom = 0.61, right = 1, top = 1, a

totalvis
```

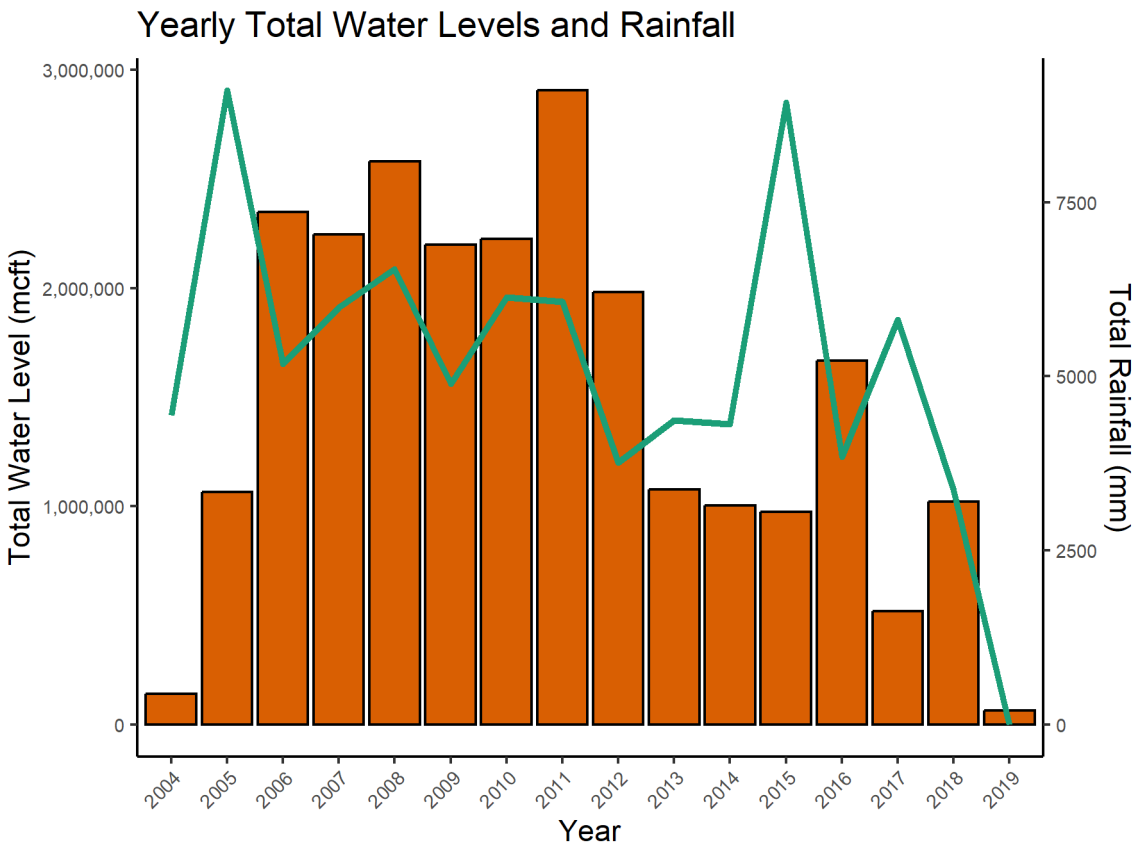



```
# Base plot using the water levels data
combined_plot <- ggplot(yearly_sum, aes(x = factor(year), y = total_level)) +
  geom_col(aes(y = total_level), fill = "#D95F02", col = "black") +
  scale_y_continuous(name = "Total Water Level (mcft)", labels = comma) +
  theme_classic() +
  xlab("Year") +
  ylab("Total Water Level (mcft)") +
  ggtitle("Yearly Total Water Levels and Rainfall") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 7),
        axis.text.y = element_text(size = 7))

# Adding the rainfall data as a line plot
max_rainfall <- max(yearly_rainfall_sum$total_rainfall, na.rm = TRUE)
scale_factor <- max(yearly_sum$total_level, na.rm = TRUE) / max_rainfall

combined_plot1 <- combined_plot +
```

```
  geom_line(data = yearly_rainfall_sum,
            aes(x = factor(year), y = total_rainfall * scale_factor, group = 1),
            color = "#1B9E77", size = 1.2) +
  scale_y_continuous(labels = comma, sec.axis = sec_axis(~./scale_factor, name = "Total Rainfall (mm)"))
```

```
## Scale for y is already present.
## Adding another scale for y, which will replace the existing scale.
```

```
combined_plot1
```



Do you see a pattern in any of these units of time?

We observe a pronounced drop in water levels, despite the rainfall's milder decline, suggesting other factors, including increased consumption, are influencing the change.

Joining

"clevels" has water data in $mcft$, while the "crainfall" data is the amount of water collected in mm. If we want to see how the rainfall impacted the level of the reservoirs, we'll need to be able to convert between the two.

Ideally, we would have some measure of how large each reservoir's watershed is so we can convert mm to mm^3 to $mcft$. See Resources for more info.

For our purposes, the rainy hurricane season in Chennai will be marked from July through December. On average, this seems to be where most rain occurs.

- That is, 0000-07-01 to 0000-12-31 each year

To do in this chunk:

- Join the two dfs by date and reservoir
- First, see how closely rising water levels and rainfall are associated
  - Pick one year and one reservoir
  - How much does rainfall seem to affect the water level in the reservoir?
- Show rainfall for a couple of years and how much lies within our definition of "wet season"
  - Is this a fair definition?

```
#column names of my joined df = date, reservoir, level, rainfall

areas_ft2 <- c(
  poondi = 317520215,
  cholavaram = 91381687,
  redhills = 216231884,
  chembarambakkam = 262626262
)

# Join datasets
joined_df <- inner_join(crain_long, clevels_long, by = c("date", "reservoir"))

# Calculate rainfall in mcft using vectorized operations
joined_df <- joined_df %>%
  mutate(
    rainfall_mcft = rainfall * 0.00328084 * areas_ft2[reservoir] / 1e6
  )

cleaned_df <- joined_df %>%
  select(date, reservoir, level, rainfall_mcft)


cleaned_df <- cleaned_df %>%
  mutate(
    year = year(date),
    month = month(date),
    yday = yday(date)
  )

# Aggregating the data for yearly average rainfall across reservoirs in mcft
yearly_avg_rainfall_mcft <- cleaned_df %>%
  group_by(year, reservoir) %>%
  summarise(average_rainfall_mcft = mean(rainfall_mcft, na.rm = TRUE))
```

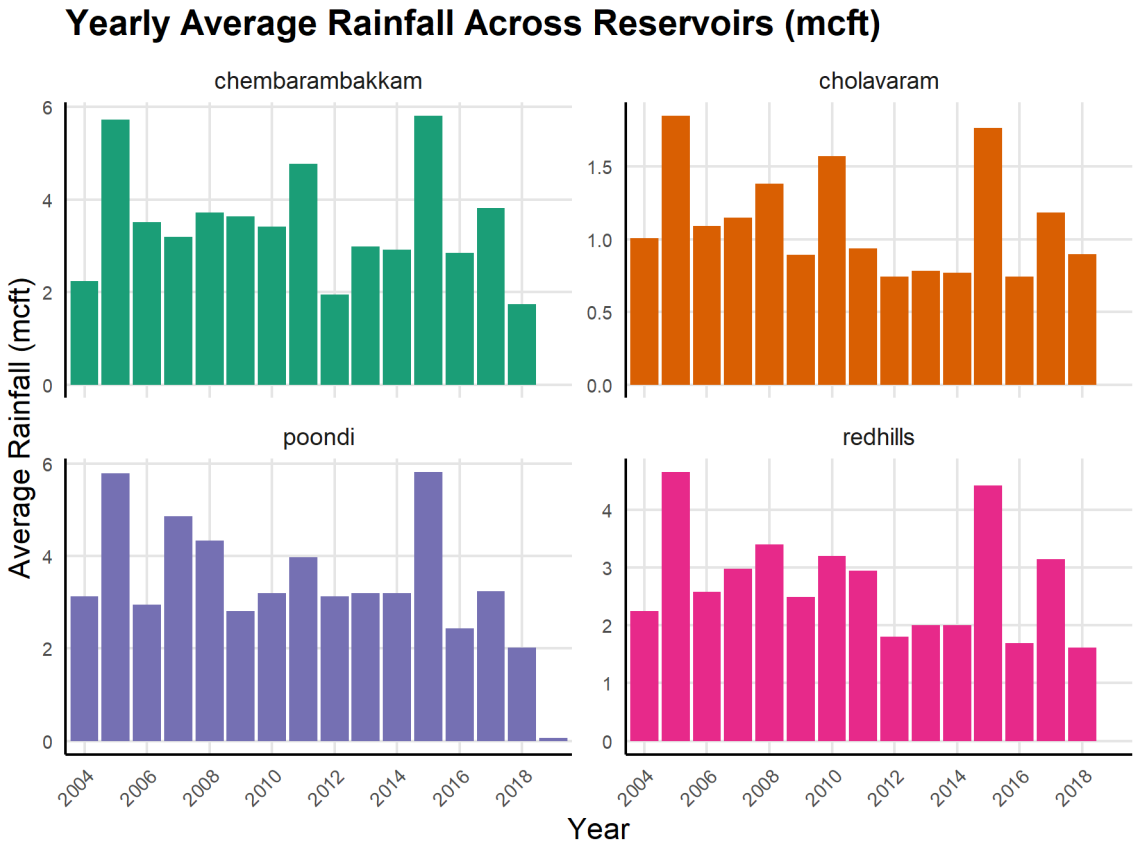
```
## `summarise()` has grouped output by 'year'. You can override using the
## `.groups` argument.
```

```
yearly_total_rainfall_mcft <- cleaned_df %>%
  group_by(year, reservoir) %>%
  summarise(total_rainfall_mcft = sum(rainfall_mcft, na.rm = TRUE))
```

```
## `summarise()` has grouped output by 'year'. You can override using the
## `.groups` argument.
```

```
ggplot(yearly_avg_rainfall_mcft, aes(x = factor(year), y = average_rainfall_mcft, fill = reservoir)) +
  geom_col() +
  facet_wrap(~ reservoir, scales = "free_y") +
  ggtitle("Yearly Average Rainfall Across Reservoirs (mcft)") +
  xlab("Year") +
```

```
  ylab("Average Rainfall (mcft)") +
  first_custom() +
  scale_x_discrete(breaks = seq(2004, 2019, by = 2)) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 7),
        axis.text.y = element_text(size = 7),
        legend.position = "none") +
  scale_fill_brewer(palette = "Dark2")
```

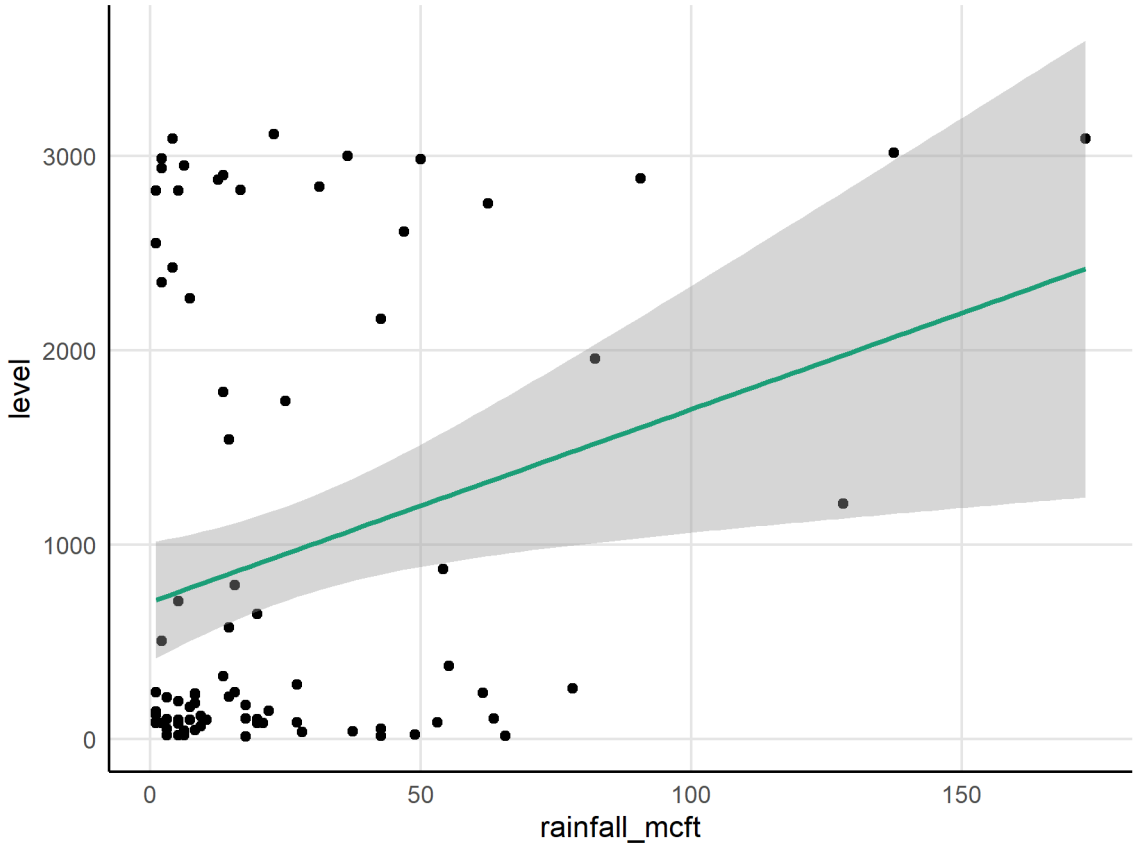



```
subset_df <- cleaned_df %>%
  filter(year == 2005 & reservoir == "poondi")

subset_df_filtered <- subset_df %>%
  filter(rainfall_mcft > 0)

correlation <- cor(subset_df_filtered$rainfall_mcft, subset_df_filtered$level)
correlation
```

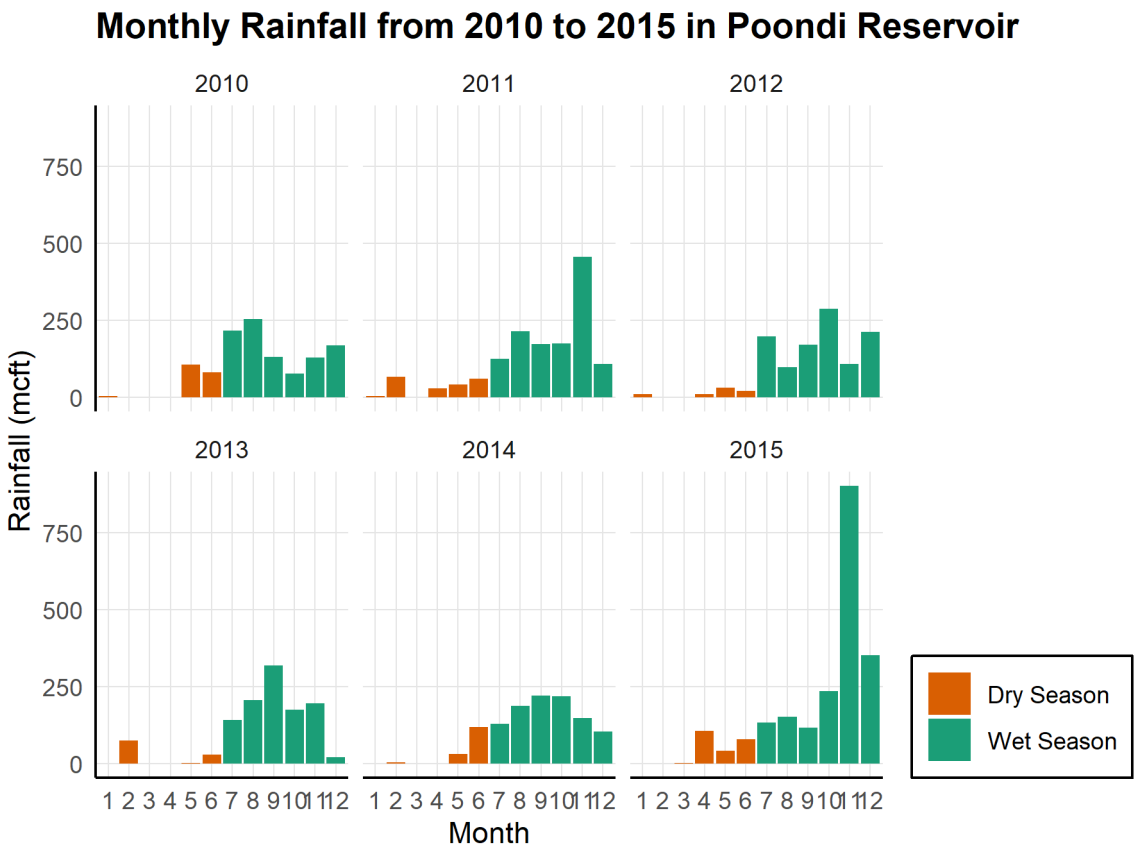
```
## [1] 0.2651515
```

```
ggplot(subset_df_filtered, aes(x=rainfall_mcft, y=level)) +
  geom_point() +
  geom_smooth(method='lm', color = "#1B9E77") +
  first_custom()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
# Subseting the data for six years (2010 to 2015) and the reservoir of interest
subset_df_6yrs <- cleaned_df %>%
  filter(year >= 2010 & year <= 2015 & reservoir == "poondi")

# Adding a column to indicate whether a month falls in the wet season or not
subset_df_6yrs <- subset_df_6yrs %>%
  mutate(wet_season = ifelse(month >= 7 & month <= 12, "Wet Season", "Dry Season"))


ggplot(subset_df_6yrs, aes(x=factor(month), y=rainfall_mcft, fill=wet_season)) +
  geom_bar(stat="identity") +
  facet_wrap(~year) +
  scale_fill_manual(values=c("Dry Season"="#D95F02", "Wet Season"="#1B9E77")) +
  labs(title="Monthly Rainfall from 2010 to 2015 in Poondi Reservoir",
       x="Month",
       y="Rainfall (mcft)") +
  first_custom() +
  theme(panel.grid.major = element_line(size = 0.3),
        legend.title = element_blank())
```



RESERVOIR	Full Tank Level (ft.)	Full Capacity (mcft)	Level (ft)	Storage (mcft)	Storage Level (%)	Inflow (cusecs)	Outflow (cusecs)	Rainfall (mm)	Storage as on same day last year (mcft)
POONDI	140.00	3231.00	136.29	2054.00	63.57	90.00	545.00	0.00	785.00
CHOLAVARAM	65.50	1081.00	59.93	572.00	52.91	102.00	10.00	0.00	184.00
PUZHAL	50.20	3300.00	46.75	2554.00	77.39	270.00	189.00	0.00	2516.00
KANNANKOTTAI THERVOY KANDIGAI	115.35	500.00	113.80	448.00	89.60	-	15.00	0.00	500.00
CHEMBARAMBAKKAM	85.40	3645.00	83.42	3125.00	85.73	153.00	188.00	0.00	2710.00
VEERANAM	47.50	1465.00	42.35	400.00	27.30	-	285.00	0.00	978.20
TOTAL	**-**	**13,222.00**	**-**	**9,153.00**	**69.23**	**615.00**	**1,232.00**	**0.00**	**7,673.20**

Figure 1: Lake Levels

I found a Lake level information on the official website : https://cmwssb.tn.gov.in/lake-level and calculated estimate areas of the reservoirs that effects the water level. Poondi: 317,520,215 ft2 Cholavaram: 91,381,687 ft2 Red Hills: 216,231,884 ft2 Chembarambakkam: 262,626,262 ft2

0.2651515 correlation coefficient suggests a weak positive relationship between the two variables. This would imply that rainfall does contribute to the reservoir's water level but the immediate effect of rainfall on reservoir levels may not be instantaneous, and could potentially manifest after some delay. There might be

additional variables or complications affecting the water level.

Based on the visual analysis of the bar graphs showing monthly rainfall for the years 2010 to 2015 in the Poondi Reservoir, it is evident that a significant portion of the annual rainfall occurs during the defined "wet season" of July to December. The color-coded bars clearly show that the rainfall during these months is consistently higher across the selected years, compared to the "dry season" months.

Factors

It's important to consider factors that affect the inflow and outflow of any water system. If the inflow is greater than the outflow, the level of water will rise, and vice-versa. A coating was installed on the bottom, so these reservoirs can lose major amounts of water in two ways:

- Flow to the city
- or evaporation

They can gain water in two ways:

- rainfall directly onto the lake or the surrounding watershed
- and aqueducts

Can you tell which factors contribute to the inflow of water into each reservoir?

The lake may receive direct rainfall or runoff from nearby watershed areas that eventually enters the reservoir.

Another possible source of inflow into the reservoir is aqueducts, which are man-made canals that are normally used to transport water from one place to another.

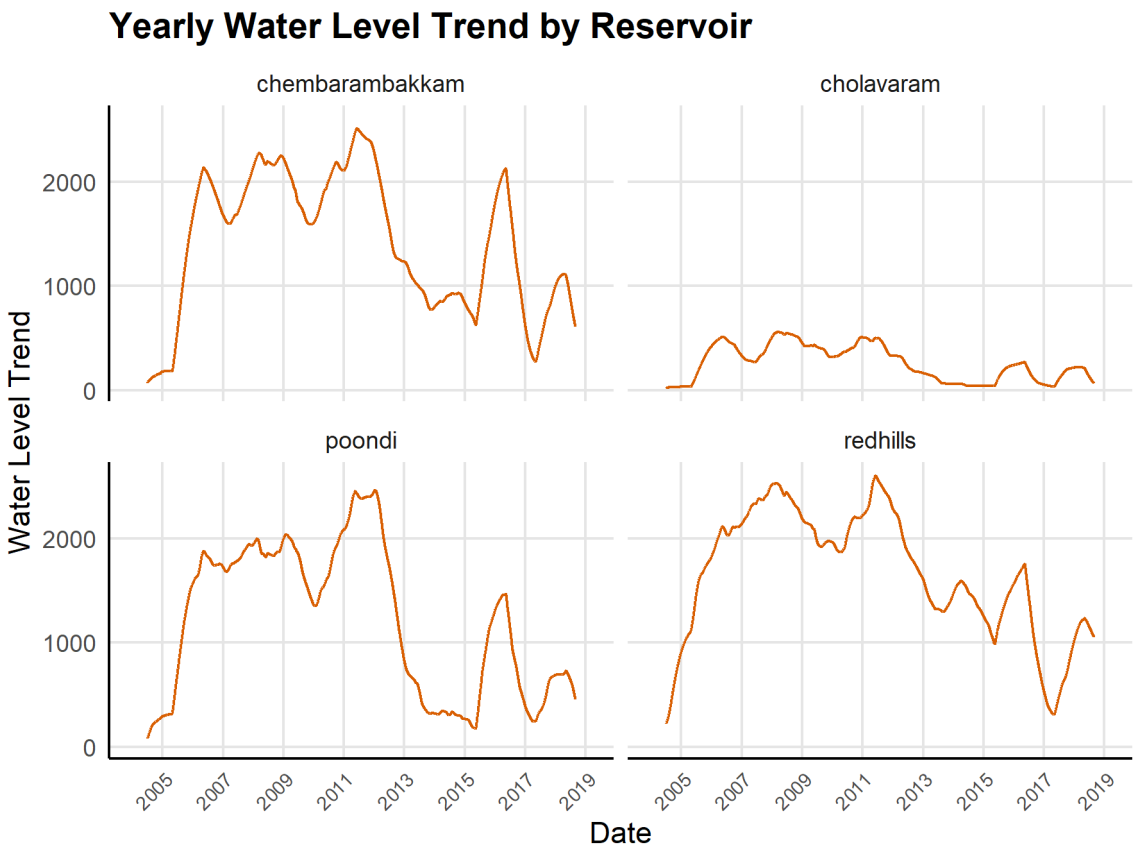
Timeseries

These are a great way to get rid of some noise in timed data. When we apply **ts()** we transform the data to a version that has dampened the effects of trends that repeat for whatever frequency we select, which makes it easier to analyse and visualize data that has hourly, monthly, or yearly patterns. Remember to consider the trends each reservoir in and out of wet season, and especially consider their combined water level!

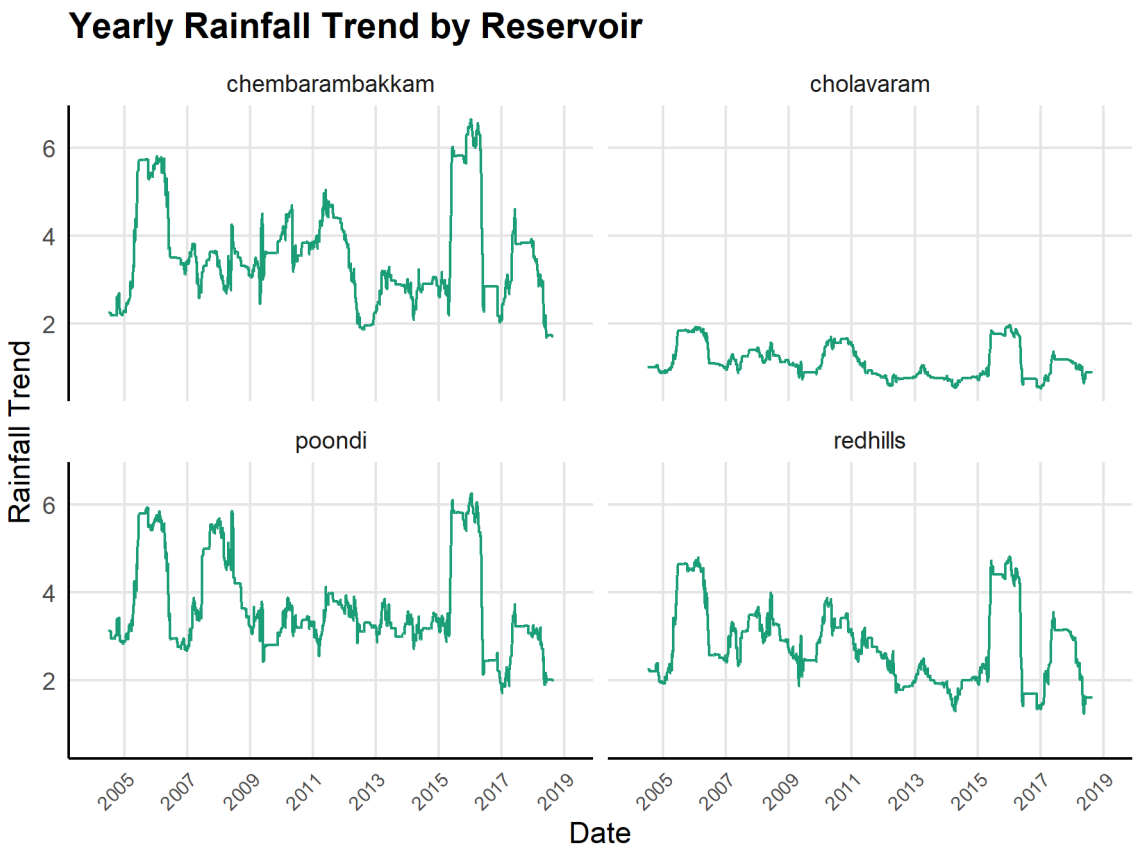
Things to do in this chunk:

- Create a new column that has trends for each reservoir and water variable
  - You'll need to group by reservoir
  - Arrange by date
  - And apply **decompose(ts())$trend** with frequency = 365 for a yearly trend
- Perform some more EDA on this dataset

```
library(patchwork)
grouped_df <- cleaned_df %>%
  group_by(reservoir) %>%
  arrange(date) %>%
  mutate(level_trend = decompose(ts(level, frequency = 365))$trend)

ggplot(grouped_df, aes(x=date, y=level_trend)) +
  geom_line(color = "#D95F02") +
  facet_wrap(~reservoir) +
  ggtitle("Yearly Water Level Trend by Reservoir") +
  xlab("Date") +
  ylab("Water Level Trend") +
  first_custom() +
  scale_x_date(date_breaks = "2 years", date_labels = "%Y") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 7))
```

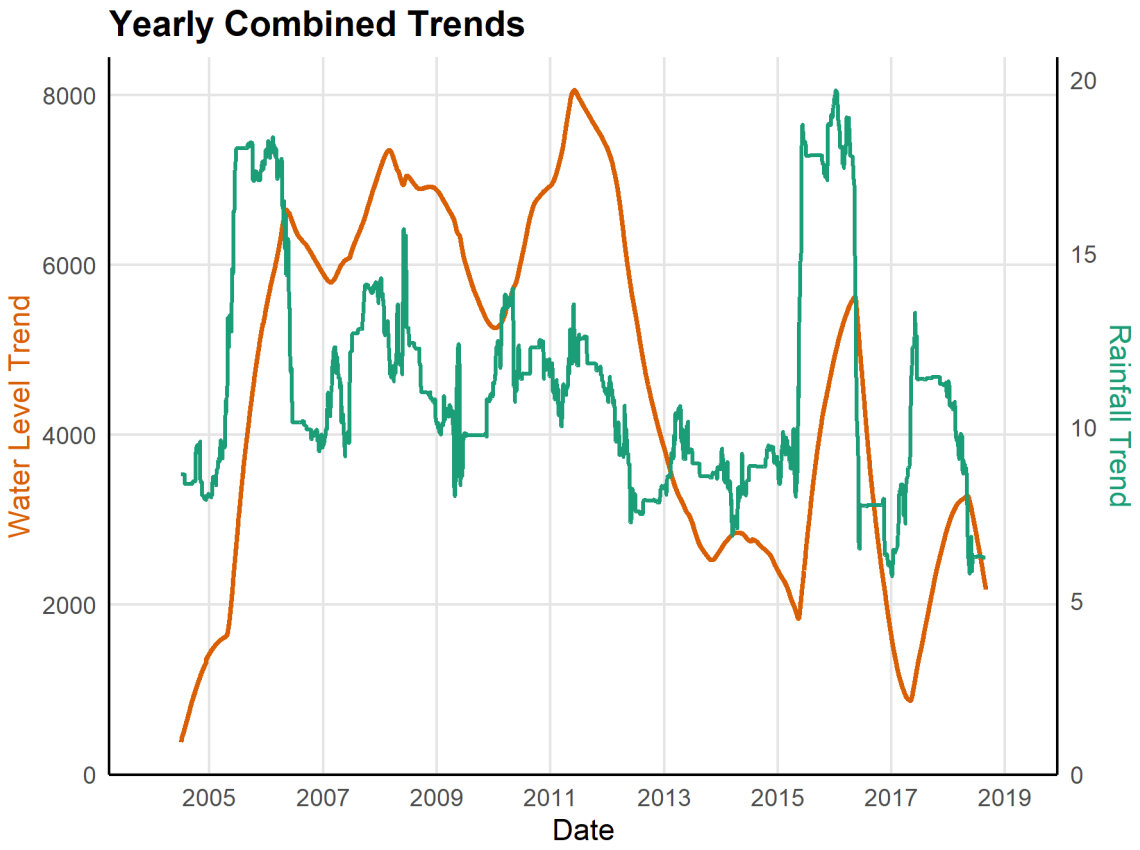



```
grouped_df <- grouped_df %>%
  group_by(reservoir) %>%
  mutate(rainfall_trend = decompose(ts(rainfall_mcft, frequency = 365))$trend)

ggplot(grouped_df, aes(x=date, y=rainfall_trend)) +
  geom_line(color = "#1B9E77") +
  facet_wrap(~reservoir) +
  ggtitle("Yearly Rainfall Trend by Reservoir") +
  xlab("Date") +
  ylab("Rainfall Trend") +
  first_custom() +
  scale_x_date(date_breaks = "2 years", date_labels = "%Y") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 7))
```




```
combined_df <- grouped_df %>%
  group_by(date) %>%
  summarise(
    combined_level = sum(level, na.rm = TRUE),
    combined_rainfall = sum(rainfall_mcft, na.rm = TRUE)
  )

combined_df <- combined_df %>%
  mutate(
    combined_level_trend = decompose(ts(combined_level, frequency = 365))$trend,
    combined_rainfall_trend = decompose(ts(combined_rainfall, frequency = 365))$trend
  )

level_max <- max(combined_df$combined_level_trend, na.rm = TRUE)
rainfall_max <- max(combined_df$combined_rainfall_trend, na.rm = TRUE)
ratio <- level_max / rainfall_max


colors_dark2 <- brewer.pal(8, "Dark2")[1:2]

yearlycombinedvis <- ggplot(data = combined_df) +
  geom_line(aes(x = date, y = combined_level_trend, color = "Water Level Trend"), linewidth = 0.8) +
  geom_line(aes(x = date, y = combined_rainfall_trend * ratio, color = "Rainfall Trend"), linewidth = 0
  scale_y_continuous(name = "Water Level Trend",
                     sec.axis = sec_axis(~./ratio, name="Rainfall Trend")) +
  ggtitle("Yearly Combined Trends") +
```

```
  xlab("Date") +
  first_custom() +
  scale_color_manual(values = colors_dark2) +
  theme(axis.title.y = element_text(color = colors_dark2[2]),
        axis.title.y.right = element_text(color = colors_dark2[1]),
        legend.position = "none") +
  scale_x_date(date_breaks = "2 years", date_labels = "%Y")

yearlycombinedvis
```



What do you see that's interesting?

The reservoir "chembarambakkam" and "poondi" both seem to have experienced their peak water levels around the same time, suggesting possible correlated factors affecting their water levels.The reservoir "redhills" shows a substantial decline in water levels starting from around 2012 onwards, which might raise concerns about water sustainability in that region. Around 2014-2015, there was a challenging period where "Cholavaram" had no water, and "Poondi" had very little. However, the presence of water in the other two reservoirs might have alleviated the situation. For the combined waterfall trends, we observe a decline from 2008 until 2015, leading to a significant decrease in water levels.

Are any trends more meaningful than others?

The yearly combined trends plot shows that the water level trend line began to clearly decline after 2013, falling below the rainfall trend line. This significant disparity may indicate things like increased use because of maybe population increase, modifications to water management procedures, or possible leaks that require further examination.

Assembly

Now that you have a good understanding of the data, gather all this knowledge and put it on a 1-page report to city officials. You need the following elements:

- Title
- Subtitle
- Author
- Date the report was completed
- 2 text grobs or annotations
- 4-6 visualizations
  - including at least 1 table
  - and 4 different varieties of plots
  - If you like the ones your already made, please use them!
- and a conclusion at the bottom of your report that either suggests no change in water management, or expresses a need for change.

Create Visualizations

Some plots you could try:

- dumbell plot
- ridgeline
- tree diagram

And don't forget to reduce clutter. Since the audience is seeing a lot of information at once, how can you minimize clutter?

- If you color by reservoir, can you use the same colors each time?
  - to make it easier to have a combined legend in the next section
- Using title vs. axis labels

Remember the goal here is to show:

- how patterns of rainfall in and out of monsoon season have changed over time
- the amount of water used by the city
- how much rainfall is needed to survive each off-season
- how much water each reservoir holds at the time of your report (which again, is March 2019)

```
monsoon_start_month <- 7
monsoon_end_month <- 12

crain_aggregated <- crain_long %>%
  group_by(year, month) %>%
  summarize(monthly_rainfall = sum(rainfall, na.rm = TRUE)) %>%
  mutate(season = ifelse(month >= monsoon_start_month & month <= monsoon_end_month, "Monsoon", "Out_of_
  group_by(year, season) %>%
  summarize(total_rainfall = sum(monthly_rainfall, na.rm = TRUE))
```

```
## `summarise()` has grouped output by 'year'. You can override using the
## `.groups` argument.
## `summarise()` has grouped output by 'year'. You can override using the
## `.groups` argument.
```

```
crain_filtered <- crain_aggregated %>%
  filter(year != 2019) %>%
  filter(year != 2004)
```

```
crain_wide <- crain_filtered %>%
  spread(key = season, value = total_rainfall)

inandoutmansoon <- ggplot(crain_wide, aes(y = factor(year))) +
  geom_segment(aes(x = Monsoon, xend = Out_of_Monsoon, yend = factor(year)), color = "#aeb6bf", size = 
  geom_point(aes(x = Monsoon, color = "Monsoon"), size = 3) +
  geom_point(aes(x = Out_of_Monsoon, color = "Out of Monsoon"), size = 3) +
  scale_color_manual(values = c("Monsoon" = "#1B9E77", "Out of Monsoon" = "#D95F02"), name = "Season") +
  labs(title = "Comparison of Total Rainfall In and Out of Monsoon", x = "Total Rainfall (mm)") +
  first_custom() +
  theme(axis.title.y = element_blank())

cleaned_df2 <- cleaned_df %>%
  group_by(date) %>%
  mutate(total_water_for_day = sum(level + rainfall_mcft)) %>%
  ungroup()

cleaned_df_summarized <- cleaned_df2 %>%
  group_by(date, year, month, yday) %>%
  summarise(
    total_level_for_day = sum(level),
    total_rainfall_for_day = sum(rainfall_mcft),
    total_water_for_day = first(total_water_for_day)
  ) %>%
  ungroup()
```

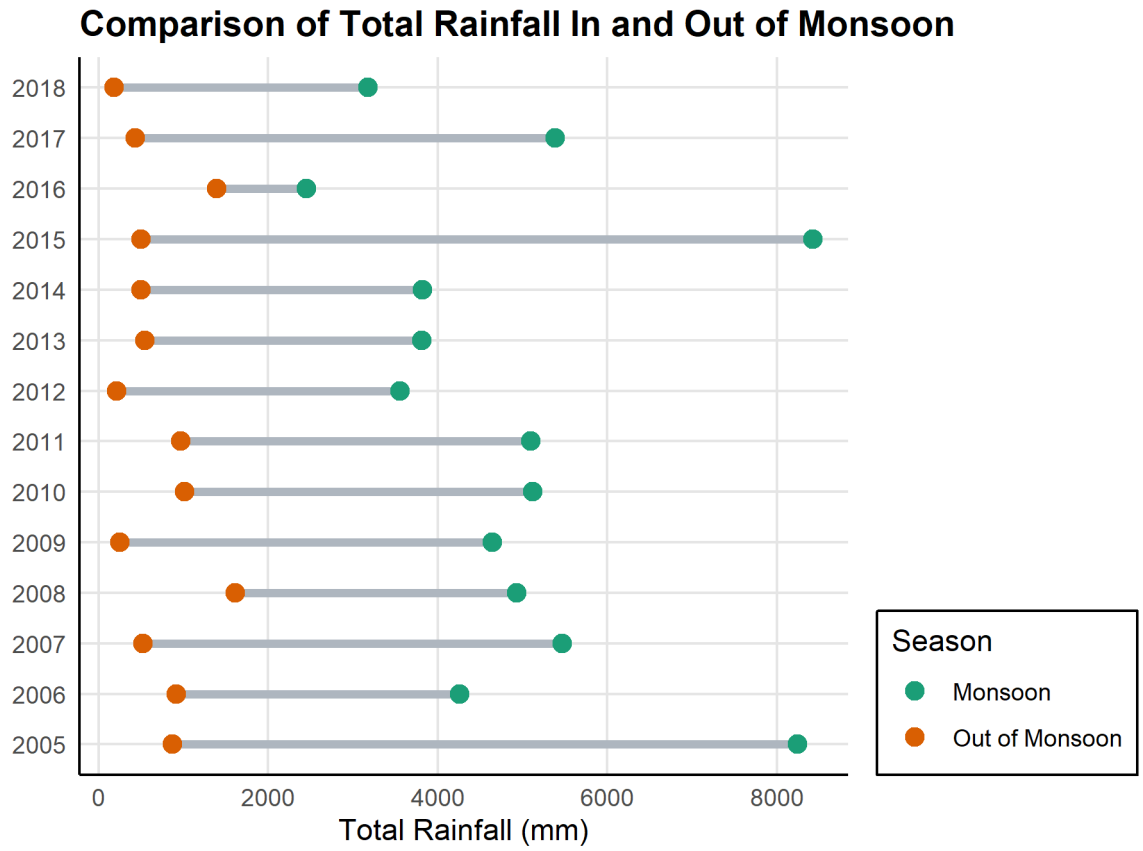
```
## `summarise()` has grouped output by 'date', 'year', 'month'. You can override
## using the `.groups` argument.
```

```
cleaned_df_summarized <- cleaned_df_summarized %>%
  arrange(date) %>%
  mutate(
    previous_total_water = lag(total_water_for_day, default = first(total_water_for_day)),
    daily_usage = previous_total_water - total_level_for_day
  )

yearly_usage_df <- cleaned_df_summarized %>%
  group_by(year) %>%
  summarise(yearly_usage = sum(daily_usage, na.rm = TRUE))

yearly_totals <- left_join(yearly_usage_df, yearly_rainfall_sum, by = "year")
yearly_totals
```

```
## # A tibble: 16 x 3
##     year yearly_usage total_rainfall
##    <dbl>        <dbl>          <dbl>
##  1  2004        1164.          4439.
##  2  2005       -1265.          9108.
##  3  2006        5581.          5181
##  4  2007        2376.          5987.
##  5  2008        5168.          6542
##  6  2009        5888.          4895.
##  7  2010        1287.          6133.
##  8  2011        5822.          6067.
```

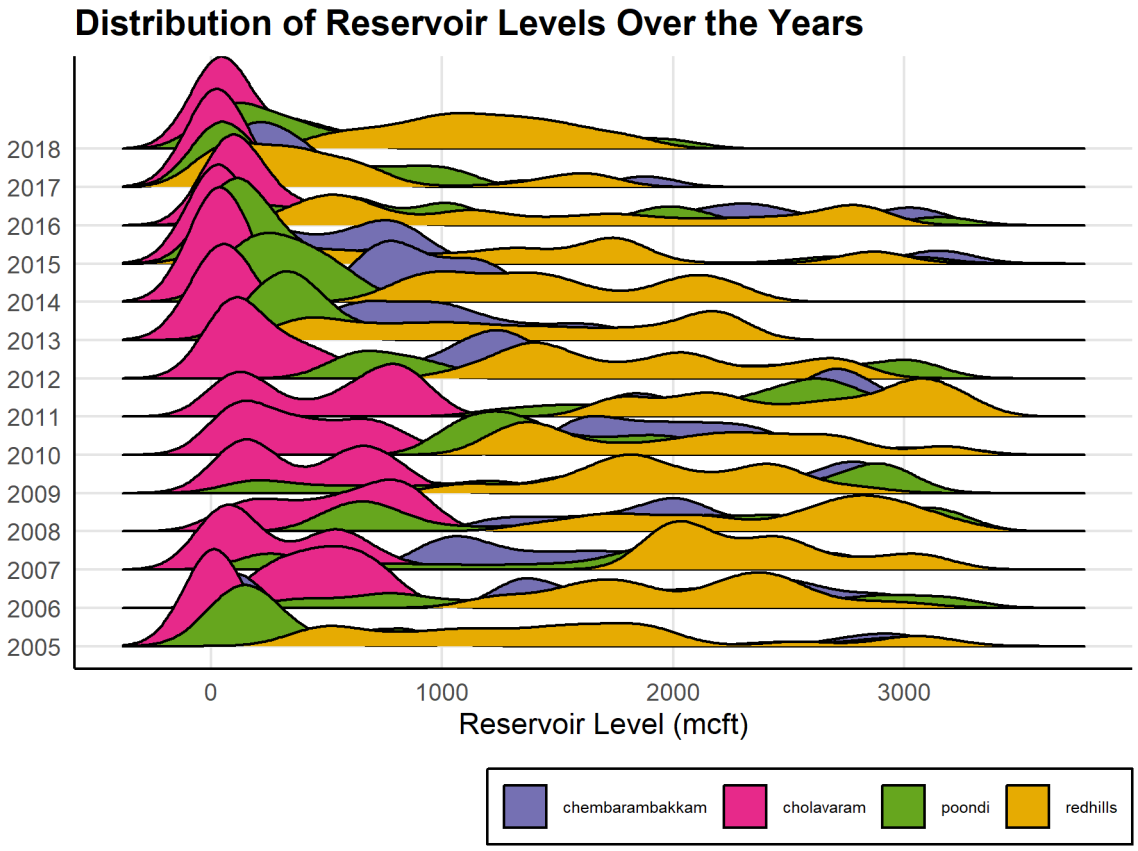
```
##  9  2012        6590.          3767.
## 10  2013        5283.          4365.
## 11  2014        3460.          4318
## 12  2015        -204.          8929.
## 13  2016       10987.          3846.
## 14  2017         895.          5814.
## 15  2018        5856.          3367
## 16  2019         334.             4
```

```
inandoutmansoon
```




```
filteredyears_data <- cleaned_df %>%
  filter(year != 2004 & year != 2019)
colors <- brewer.pal(8, "Dark2")[3:6]

ridgevis <- ggplot(filteredyears_data, aes(x = level, y = as.factor(year), fill = reservoir)) +
  geom_density_ridges(scale = 3) +
  labs(title = "Distribution of Reservoir Levels Over the Years",
       x = "Reservoir Level (mcft)", y = "Year") +
  first_custom() +
  scale_fill_manual(values = colors) +
  theme(legend.position = "bottom",
        legend.title = element_blank(),
        axis.title.y = element_blank(),
        legend.text = element_text(size = 6))

ridgevis
```

```
## Picking joint bandwidth of 128
```



To find the amount of water used by the city, I followed these steps:

```
Filtered Outliers and Low Usage Years: Excluded years like 2005 and 2015 with high rainfall and those wi

Calculated a Baseline Usage: Found the average yearly water usage from the remaining years.

Noticed Post-Rainfall Usage Increase: Observed that water usage increased the year after significant rai

Estimated Water Usage: Used the baseline and added half of the observed increase to predict usage after
```

From the data:

```
Baseline = 5283.4160
Difference = 5405.5982
Estimated Usage = 7986.2151
```

So, I estimate that after significant rainfall, the city might use around 7,986.2151 units of water.

```
library(png)
library(grid)

cleaned_df_summarized <- cleaned_df_summarized %>%
  filter(year != 2019)

off_season_data <- cleaned_df_summarized %>%
  filter(month %in% 1:6)
```

```r
off_season_usage <- off_season_data %>%
  group_by(year) %>%
  summarise(
    total_off_season_usage = sum(daily_usage, na.rm = TRUE)
  )

beginning_storage <- cleaned_df_summarized %>%
  group_by(year) %>%
  filter(row_number(desc(date)) == 1) %>%
  select(year, total_level_for_day)


colnames(beginning_storage)[2] <- "start_of_off_season_storage"


beginning_storage <- beginning_storage %>%
  mutate(year = year + 1)

beginning_storage
```

```
## # A tibble: 15 x 2
## # Groups:   year [15]
##     year start_of_off_season_storage
##    <dbl>                       <dbl>
##  1  2005                       2264.
##  2  2006                      10112
##  3  2007                       8232
##  4  2008                      10302
##  5  2009                       9831
##  6  2010                       7531
##  7  2011                      10404
##  8  2012                       9012
##  9  2013                       5385
## 10  2014                       3377
## 11  2015                       3154
## 12  2016                       9866
## 13  2017                       1706
## 14  2018                       4969
## 15  2019                       1402
```

```r
required_rainfall_data <- left_join(off_season_usage, beginning_storage, by = "year") %>%
  mutate(off_season_required_rainfall = start_of_off_season_storage - total_off_season_usage)


required_rainfall_data
```

```
## # A tibble: 15 x 4
##     year total_off_season_usage start_of_off_season_sto~1 off_season_required_~2
##    <dbl>                  <dbl>                      <dbl>                   <dbl>
##  1  2004                  1066.                        NA                      NA
##  2  2005                  1691.                      2264.                   573.
##  3  2006                  5985.                     10112                   4127.
##  4  2007                  4983.                      8232                   3249.
##  5  2008                  5790.                     10302                   4512.
```

```
##  6  2009                 5032.                   9831                 4799.
##  7  2010                 3565.                   7531                 3966.
##  8  2011                 5987.                  10404                 4417.
##  9  2012                 5600.                   9012                 3412.
## 10  2013                 4282.                   5385                 1103.
## 11  2014                 1210.                   3377                 2167.
## 12  2015                 2258.                   3154                  896.
## 13  2016                 6572.                   9866                 3294.
## 14  2017                 1934.                   1706                 -228.
## 15  2018                 2869.                   4969                 2100.
## # i abbreviated names: 1: start_of_off_season_storage,
## #   2: off_season_required_rainfall
```

```r
combined_data <- left_join(required_rainfall_data, yearly_totals, by = "year")

final_table <- combined_data %>%
  select(year, yearly_usage, total_rainfall, total_off_season_usage, off_season_required_rainfall)

final_table <- final_table %>%
  rename(
    Year = year,
    "Yearly Usage" = yearly_usage,
    "Total Rainfall" = total_rainfall,
    "Off-season Usage" = total_off_season_usage,
    "Req Off-seas Rain" = off_season_required_rainfall
  )

final_table[] <- lapply(final_table, function(x) if(is.numeric(x)) round(x, 2) else x)
final_table <- final_table %>%
  filter(Year != 2004)



# compact_table <- kable(final_table, "html", align = 'c', caption = '<b style="font-size: 14px;">Chenn
# kable_classic(full_width = F, html_font = "Cambria", font_size = 12)
#
# compact_table
# save_kable(compact_table, "my_table.html")



tableimg <- readPNG("watertable.png")
img_grob <- rasterGrob(tableimg, interpolate=TRUE)

table_ggplot <- ggplot() +
  annotation_custom(img_grob, xmin=-Inf, xmax=Inf, ymin=-Inf, ymax=Inf) +
  labs(title = "Water Usage, Rainfall, and Off-season Metrics")

table_ggplot
```

Water Usage, Rainfall, and Off-season Metrics

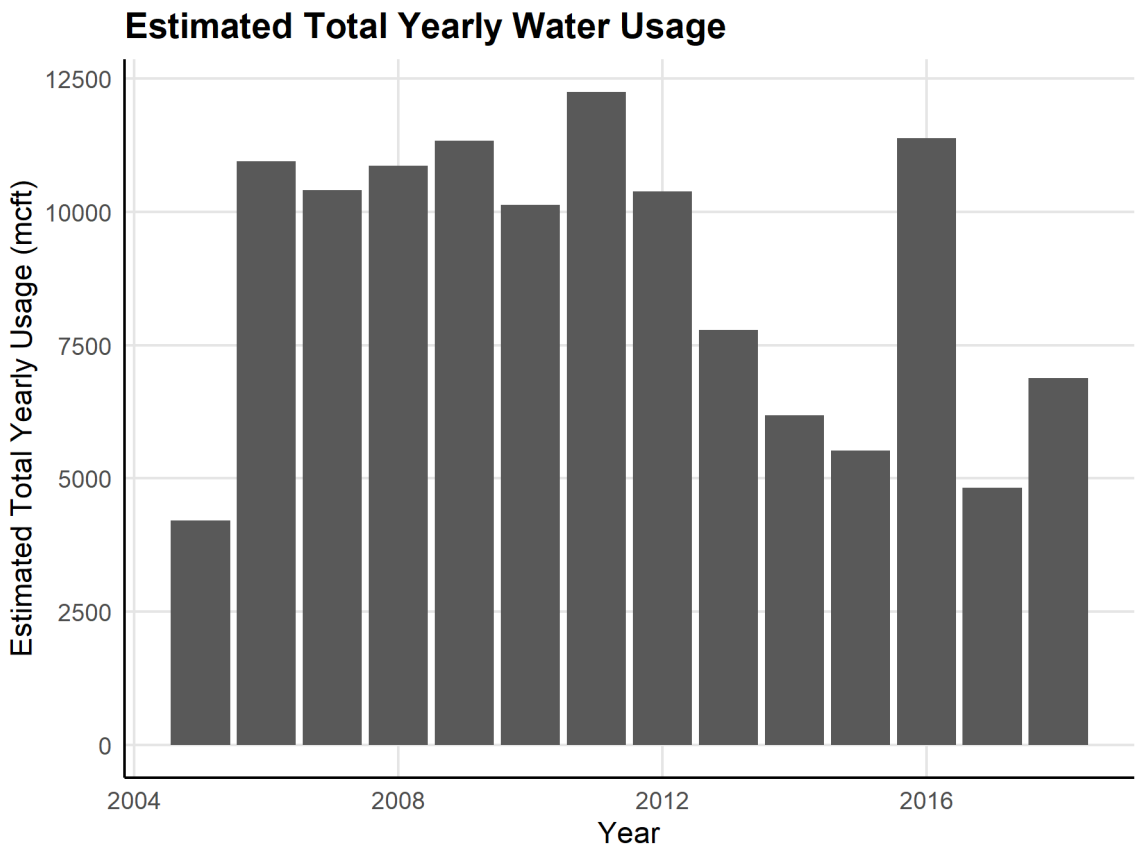
Year	Yearly Usage	Total Rainfall	Off-season Usage	Req Off-seas Rain
2005	-1265.40	9107.50	1691.01	572.89
2006	5581.16	5181.00	5985.21	4126.79
2007	2376.00	5987.40	4982.72	3249.28
2008	5168.47	6542.00	5789.82	4512.18
2009	5887.77	4894.85	5032.39	4798.61
2010	1286.87	6133.10	3565.43	3965.57
2011	5822.45	6066.80	5986.90	4417.10
2012	6590.42	3766.60	5599.75	3412.25
2013	5282.70	4364.95	4282.13	1102.87
2014	3459.56	4318.00	1209.73	2167.27
2015	-203.62	8928.70	2258.25	895.75
2016	10986.76	3845.80	6571.87	3294.13
2017	895.44	5814.40	1934.07	-228.07
2018	5856.09	3367.00	2869.25	2099.75

I analyzed the water usage of reservoirs during the off-season months (January to June). After removing incomplete data from 2019, I calculated the total off-season water usage for each year. I also determined the water levels at the beginning of each off-season. By comparing the beginning water levels with the usage, I computed the required rainfall needed to sustain reservoir levels.

I still couldn't find a way to interpret data to find the actual values clearly.

```
cleaned_df$total_level <- with(cleaned_df, ave(level, date, FUN = sum))
filtered_df <- cleaned_df %>%
  filter(!(year %in% c(2004, 2019)))

summarized_df <- filtered_df %>%
  group_by(date) %>%
  summarize(
    total_level = first(total_level),
    total_rainfall_mcft = sum(rainfall_mcft)
  ) %>%
  ungroup()

consecutive_no_rain_data <- summarized_df %>%
  arrange(date) %>%
  mutate(
    next_date = lead(date, 1),
    next_total_level = lead(total_level, 1),
    next_rainfall = lead(total_rainfall_mcft, 1),
    day_difference = as.numeric(difftime(next_date, date, units = "days"))
  ) %>%
```

```r
  filter(
    total_rainfall_mcft == 0 &
    next_rainfall == 0 &
    day_difference == 1 &
    next_total_level < total_level
  ) %>%
  select(-next_date, -day_difference)

consecutive_no_rain_data <- consecutive_no_rain_data %>%
  arrange(date) %>%
  mutate(
    prev_date = lag(date, 1),
    next_date_2 = lead(date, 1),
    connected_before = as.numeric(difftime(date, prev_date, units = "days")) == 1,
    connected_after = as.numeric(difftime(next_date_2, date, units = "days")) == 1
  ) %>%
  filter(connected_before | connected_after) %>%
  select(-connected_before, -connected_after, -prev_date, -next_date_2)

consecutive_no_rain_data <- consecutive_no_rain_data %>%
  mutate(daily_diff = total_level - next_total_level)

# Calculate the IQR
Q1 <- quantile(consecutive_no_rain_data$daily_diff, 0.25)
Q3 <- quantile(consecutive_no_rain_data$daily_diff, 0.75)
IQR <- Q3 - Q1

# Define bounds for outliers
lower_bound <- Q1 - 1.5 * IQR
upper_bound <- Q3 + 1.5 * IQR

# Filter out the outliers
consecutive_no_rain_data_cleaned <- consecutive_no_rain_data %>%
  filter(daily_diff >= lower_bound & daily_diff <= upper_bound)

daily_avg <- consecutive_no_rain_data_cleaned %>%
  group_by(year = lubridate::year(date)) %>%
  summarise(avg_daily_usage = mean(daily_diff))

estimated_yearly_usage <- daily_avg %>%
  mutate(estimated_total_yearly_usage = avg_daily_usage * 365)

ggplot(estimated_yearly_usage, aes(x=year, y=estimated_total_yearly_usage)) +
  geom_bar(stat="identity") +
  labs(title="Estimated Total Yearly Water Usage", x="Year", y="Estimated Total Yearly Usage (mcft)") +
  first_custom()
```



In this analysis, I tried something and examined water usage patterns of a city by focusing on days without rainfall, as these days reflect true consumption from stored water. I calculated daily water usage by observing the drop in reservoir levels and then extrapolated this to estimate annual consumption. The chart, "Estimated Total Yearly Water Usage," shows fluctuations in water usage across years. Notably, the city's consumption is influenced by rainfall amounts: more rainfall replenishes reservoirs, potentially leading to increased usage. Conversely, limited rainfall might result in conservation efforts and reduced consumption. This relationship underscores the importance of considering rainfall when evaluating a city's water usage.

Modeling

```
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
##
## Attaching package: 'forecast'
```

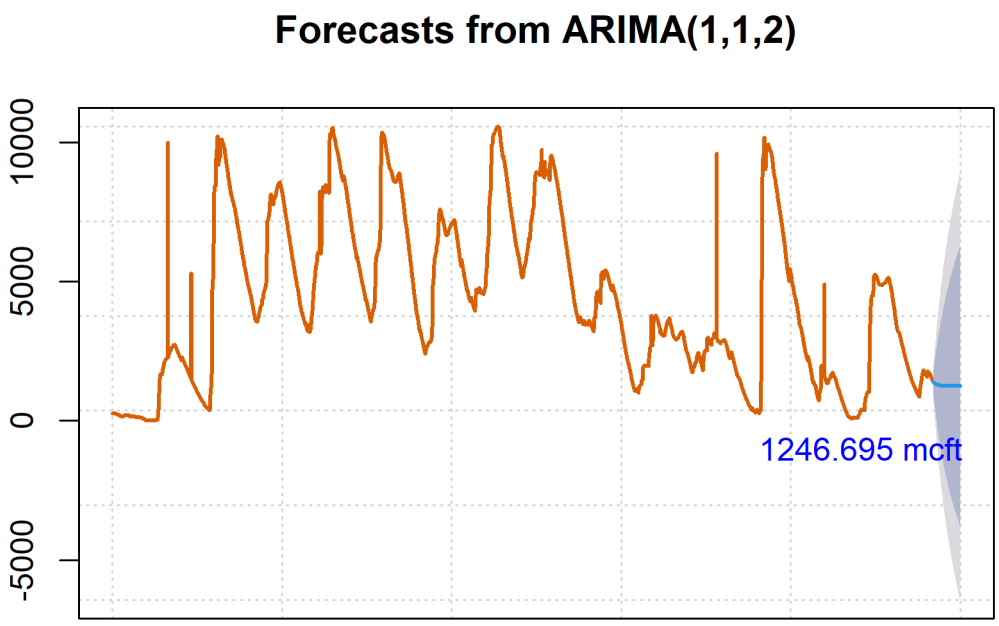
```
## The following object is masked from 'package:ggpubr':
##
##     gghistogram
```

```
library(png)

model <- auto.arima(cleaned_df_summarized$total_level_for_day)
forecast_data <- forecast(model, h=181) # 2019-07-01 is roughly half a year away from our last date in
```

```
forecast_data$mean[181]
```

```
## [1] 1246.695
```

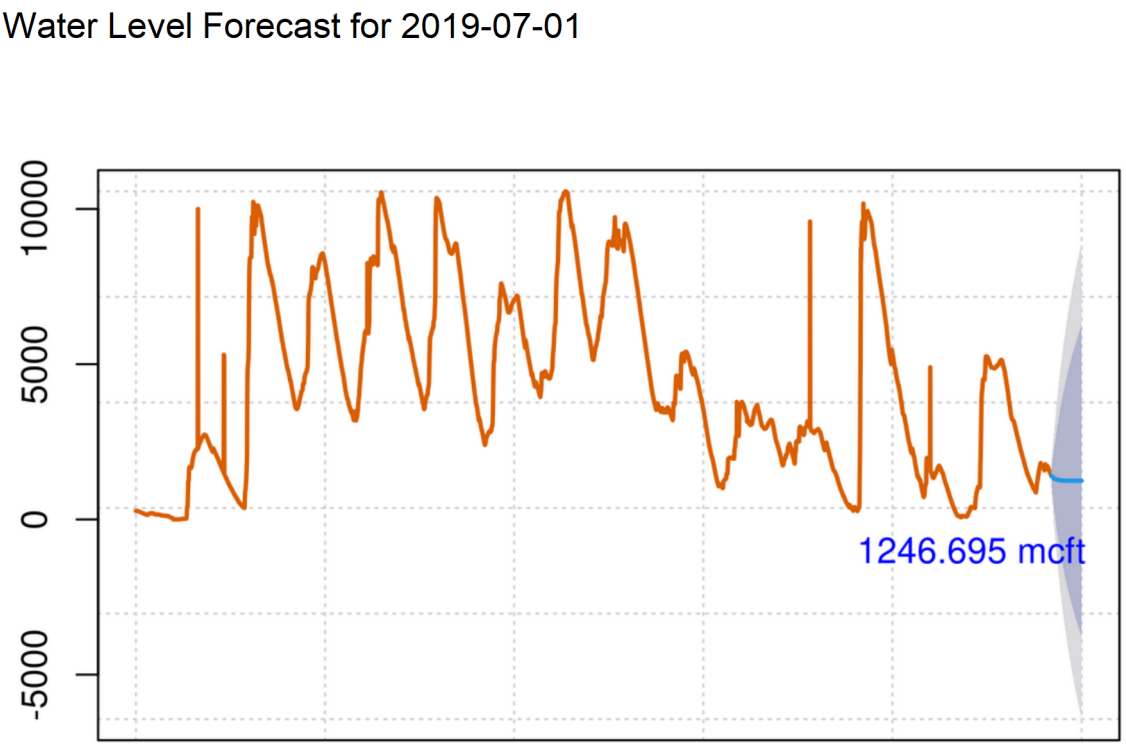
```
forecast_plot <- function() {
  plot(forecast_data,
       col="#D95F02",
       lwd=2,
       xaxt="n",
       cex.axis=0.6,
       panel.first = grid())

       text(x = +5000, y = -1000, labels = "1246.695 mcft", col = "blue", cex=1)
}

forecast_plot()
```




```
png("plot.png", width=6, height=4.5, units="in", res=300)
plot(forecast_data,
       col="#D95F02",
       lwd=2,
       xaxt="n",
       cex.axis=0.6,
       panel.first = grid())

       text(x = +5000, y = -1000, labels = "1246.695 mcft", col = "blue", cex=1)
dev.off()
```

```
## png
##   2
```

```
img <- rasterGrob(as.raster(readPNG("forecast.png")))

forecastvis <- ggplot() + annotation_custom(img, xmin=-Inf, xmax=Inf, ymin=-Inf, ymax=Inf) +
  theme_minimal() +
  labs(title = "Water Level Forecast for 2019-07-01")

forecastvis
```



Arranging

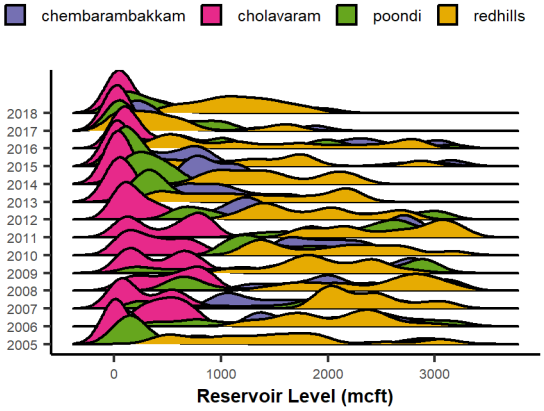
Chennai Water Supply and Demand Analysis

Quarterly Report on Water Management

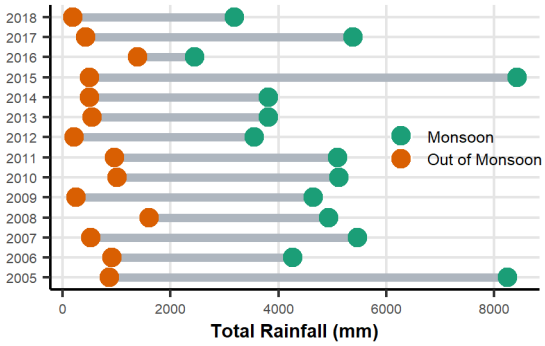
Author: Ibrahim Uruc Tarim Date: 11.02.2023

Chennai, India's significant cultural, economic, and educational hub in the south, faces water supply and demand challenges. The city, home to 10 million residents, draws its water primarily from four reservoirs with a combined capacity of 11,057 mcft. 2019 witnessed a severe water crisis, leading many to rely on wells and tanker trucks. Recent monsoon seasons have somewhat alleviated these concerns. This report provides an insight into Chennai's water management, crucial for future planning.

Distribution of Reservoir Levels Over the Years



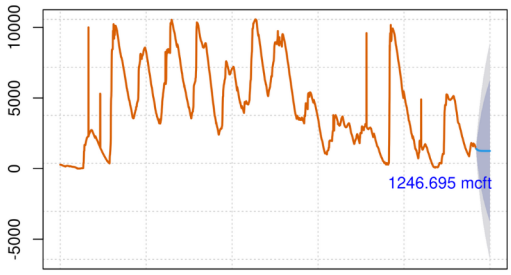
Comparison of Total Rainfall In and Out of Monsoon

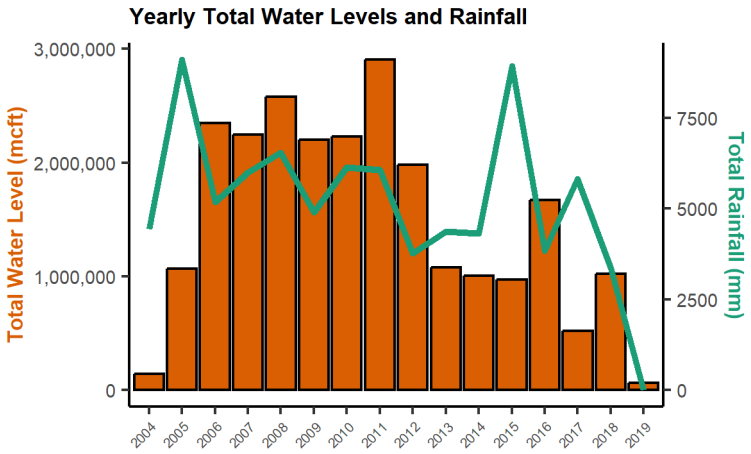


Water Usage, Rainfall, and Off-season Metrics

Year	Yearly Usage	Total Rainfall	Off-season Usage	Req Off-seas Rain
2005	-1265.40	9107.50	1691.01	572.89
2006	5581.16	5181.00	5985.21	4126.79
2007	2376.00	5987.40	4982.72	3249.28
2008	5168.47	6542.00	5789.82	4512.18
2009	5887.77	4894.85	5032.39	4798.61
2010	1286.87	6133.10	3565.43	3965.57
2011	5822.45	6066.80	5986.90	4417.10
2012	6590.42	3766.60	5599.75	3412.25
2013	5282.70	4364.95	4282.13	1102.87
2014	3459.56	4318.00	1209.73	2167.27
2015	-203.62	8928.70	2258.25	895.75
2016	10986.76	3845.80	6571.87	3294.13
2017	895.44	5814.40	1934.07	-228.07
2018	5856.09	3367.00	2869.25	2099.75

Water Level Forecast for 2019-07-01





Rainfall Patterns: Non-monsoon rainfall has declined over the years, indicating potential dry-season vulnerabilities. Water Consumption: Despite rainfall spikes, Chennai's water usage remains high, underscoring the need for efficient management.Negative Reservoir Values: The data indicates that in certain years, such as 2017, the reservoir had an excess of water beyond the city's usage. Estimated Water Usage: After significant rainfall, it's estimated that the city might consume approximately 8000 mcft units of water. Off-season Dependency: Data shows the city's reliance on rainfall to sustain reservoir levels, with occasional years having excess reserves. Reservoir Status: Future Levels Warning: Forecasting suggests a potential water crisis if the current consumption trend persists without substantial rainfall.We see a significant water level drop, even with slight rainfall decline, indicating factors like rising consumption due to population growth.

To better manage water resources, it's essential for the city to keep an eye on consumption patterns, especially after heavy rainfalls. Given the excess water in certain years, the city might consider investing in additional storage facilities or exploring ways to optimize water distribution. Furthermore, promoting water conservation methods during peak usage times can help in ensuring a more consistent water supply throughout the year.

Resources

Links:

- Date Values in R
- [Inflow (hydrology) WIkipedia](https://en.wikipedia.org/wiki/Inflow_(hydrology)
- NOAA What is a Watershed?
- Documentation for ggraph
- ggarrange and more
- R Graph Gallery
- Plot your fitted model
- Fundamentals of Data Visualization, Claus Wilke