Lab 1: Climate Data Collection and Processing

Regional Climate Trends Project

Learning R Through Climate Science

EA050

January 23, 2026 v. 0.07

Contents

1 Introduction

Welcome to Lab 1 of the Regional Climate Trends Project! This lab is designed to teach you fundamental R programming concepts while analyzing real climate data. You will learn:

- How R handles dates and time
- Data manipulation with subsetting and aggregation
- Creating basic plots
- Understanding data structures (data frames, lists)
- Working with NOAA climate data

1.1 Learning Objectives

By the end of this lab, you will be able to:

1. Convert between date formats in R
2. Subset data frames by conditions
3. Aggregate data using the `aggregate()` function
4. Create basic time series plots
5. Understand anomalies and their calculation

2 Setup

First, we need to set up our R environment. The following code loads all necessary packages and functions.

```
##
## ============================================================
##  Climate Narratives Functions v7.0 Loaded Successfully!
## ============================================================
##  Run check_packages() to verify dependencies
## ============================================================
##
##  QUICK START:
##  1. setup_project('CA')
##  2. select_stations_for_analysis(n_stations = 50)
##  3. download_stations()
##  4. load_and_save_stations(cleanup = TRUE)
##  5. process_all_stations_for_spatial()
##  6. create_spatial_objects(all_station_trends)
```

```
##
##  NEW IN v7.0:
##  - Fixed figuresfolder variable handling
##  - Improved error messages
##  - Better documentation for teaching
##
## ============================================================
```

```
# Set working directory to your project folder
# IMPORTANT: Change this path to match your computer!
setwd("/path/to/your/project/folder/")

# Load the consolidated functions file
source("ClimateNarrativesFunctions_v07.R")

# Check that all required packages are installed
check_packages()
```

R Concept: Working Directory

The `setwd()` function sets your "working directory" – the folder where R looks for files by default. Always set this at the beginning of your script to ensure file paths work correctly.

3 Project Initialization

Let's set up the project structure and download station information.

```
##
## ============================================================
##   Climate Narratives Project Setup v7.0
## ============================================================
##   Enhanced for improved spatial analysis
## ============================================================
##   Project path: /home/mwl04747/RTricks/05_Regional_Climate_Trends/2026_7
## ============================================================
##
## Creating directories...
##   [OK] Exists: /home/mwl04747/RTricks/05_Regional_Climate_Trends/2026_7/Data
##   [OK] Exists: /home/mwl04747/RTricks/05_Regional_Climate_Trends/2026_7/Output
##   [OK] Exists: /home/mwl04747/RTricks/05_Regional_Climate_Trends/2026_7/Figures
##
## [OK] Set folder variables:
```

```
##      datafolder    = /home/mwl04747/RTricks/05_Regional_Climate_Trends/2026_7/Data/
##      figuresfolder = /home/mwl04747/RTricks/05_Regional_Climate_Trends/2026_7/Figures/
##
## [OK] Station inventory already exists
##
## [OK] Found 614 potential stations for CA
##
## ============================================================
##   Setup Complete!
## ============================================================
##   Variables created in global environment:
##     * my.state       = CA
##     * my.inventory   = 614 potential stations
##     * datafolder     = /home/mwl04747/RTricks/05_Regional_Climate_Trends/2026_7/Data/
##     * figuresfolder  = /home/mwl04747/RTricks/05_Regional_Climate_Trends/2026_7/Figures/
## ============================================================
##   Next step:
##     select_stations_for_analysis(n_stations = 50)
## ============================================================
```

```
# Set your state (change this to your assigned state!)
my.state <- "CA"

# Set up project folders and download station inventory
setup_project(my.state)
```

R Concept: Variable Assignment

In R, we use `<-` to assign values to variables. The variable name goes on the left, and the value goes on the right. You can also use `=`, but `<-` is the traditional R style.

The `setup_project()` function creates several important variables in your R environment:

- `my.state` – Your two-letter state code
- `my.inventory` – Data frame of available weather stations
- `datafolder` – Path to the Data folder
- `figuresfolder` – Path to the Figures folder

4 Station Selection

Now we select high-quality stations with long data records.

```
## ============================================================
##   Selecting Stations for Analysis v7.0
## ============================================================
## Quality filters:
##   Minimum record length: 50 years
##   Must have data through: 2020
##   Target number of stations: 50
##
## Stations meeting quality criteria: 223
## Selected top 50 stations by record length
## ============================================================
## Selected Station Summary:
## ============================================================
##   Number of stations: 50
##   Oldest station start: 1870
##   Newest station start: 1906
##   Average record length: 129.4 years
##   Median record length: 129 years
## ============================================================
## [OK] Updated my.inventory with 50 selected stations
## [OK] Saved to: /home/mwl04747/RTricks/05_Regional_Climate_Trends/2026_7/Data/selected_inv
##
## Next step: download_stations()
```

```
# Select 50 stations with quality filters
selected_inventory <- select_stations_for_analysis(
  n_stations = 50,    # Number of stations to select
  min_years = 50,     # Minimum years of data required
  min_last_year = 2020 # Must have data through this year
)
```

R Concept: Function Arguments

Functions in R can have multiple **arguments** (inputs). You can specify them by name (like `n_stations = 50`) or by position. Using names makes your code clearer and helps avoid mistakes.

Let's examine the selected stations:

```
##            ID                  NAME FIRSTYEAR LASTYEAR RECORD_LENGTH
## 1 USC00043157           FT BIDWELL      1870     2026           157
## 2 USW00023271     SACRAMENTO 5 ESE      1877     2025           149
## 3 USC00042294 DAVIS 2 WSW EXP FARM      1893     2026           134
## 4 USC00046074  NAPA STATE HOSPITAL      1893     2026           134
```

```
## 5 USC00046136          NEVADA CITY      1893     2026          134
## 6 USC00046719             PASADENA      1893     2026          134
```

```
# View the first 6 rows of our station inventory
head(selected_inventory[, c("ID", "NAME", "FIRSTYEAR",
                            "LASTYEAR", "RECORD_LENGTH")])
```

R Concept: Subsetting Data Frames

Data frames are like spreadsheets – they have rows and columns. We access specific parts using square brackets: `df[rows, columns]`.

- `df[1, ]` – First row, all columns
- `df[, 1]` – All rows, first column
- `df[1:5, c("col1", "col2")]` – Rows 1-5, specific columns
- `head(df)` – First 6 rows (convenient shortcut)

5 Download Station Data

Note: This step downloads data from NOAA and takes 10-30 minutes. Run it once, then the data is saved locally.

```
# Download data from NOAA (run once, takes 10-30 minutes)
download_stations()
```

```
# Download data from NOAA (run once, takes 10-30 minutes)
download_stations()
```

6 Load and Save Data

After downloading, we load the data into R and save it in an efficient format.

```
## ============================================================
##   Loading and Processing Station Data
## ============================================================
## [1/50] USC00043157 (228,582 records)
```

```
## [2/50] USW00023271 (196,575 records)
## [3/50] USC00042294 (364,733 records)
## [4/50] USC00046074 (211,071 records)
## [5/50] USC00046136 (245,445 records)
## [6/50] USC00046719 (218,343 records)
## [7/50] USC00046826 (200,734 records)
## [8/50] USC00047821 (195,308 records)
## [9/50] USC00047902 (231,585 records)
## [10/50] USC00049866 (219,335 records)
## [11/50] USC00040693 (220,819 records)
## [12/50] USC00044412 (73,202 records)
## [13/50] USC00048351 (150,489 records)
## [14/50] USC00041614 (216,044 records)
## [15/50] USC00046730 (211,996 records)
## [16/50] USC00047880 (150,982 records)
## [17/50] USC00044259 (227,308 records)
## [18/50] USC00049073 (147,356 records)
## [19/50] USC00043161 (182,043 records)
## [20/50] USC00047195 (212,706 records)
## [21/50] USC00048702 (188,081 records)
## [22/50] USW00023157 (315,977 records)
## [23/50] USC00040943 (134,213 records)
## [24/50] USC00043875 (205,387 records)
## [25/50] USC00042805 (196,827 records)
## [26/50] USC00043191 (174,331 records)
## [27/50] USC00048839 (191,648 records)
## [28/50] USC00049367 (228,008 records)
## [29/50] USC00043747 (221,740 records)
## [30/50] USC00044890 (215,181 records)
## [31/50] USC00042239 (216,876 records)
## [32/50] USC00045532 (208,262 records)
## [33/50] USC00049452 (207,388 records)
## [34/50] USC00041018 (191,794 records)
## [35/50] USC00044223 (200,261 records)
## [36/50] USC00044500 (209,634 records)
## [37/50] USC00047965 (200,730 records)
## [38/50] USC00046168 (205,158 records)
## [39/50] USC00044997 (210,787 records)
## [40/50] USC00046508 (200,300 records)
## [41/50] USC00046506 (214,291 records)
## [42/50] USC00046252 (186,788 records)
## [43/50] USC00040383 (197,382 records)
## [44/50] USC00041912 (201,996 records)
## [45/50] USC00046399 (204,277 records)
## [46/50] USC00049855 (204,181 records)
```

```
## [47/50] USC00048353 (218,776 records)
## [48/50] USC00049699 (215,085 records)
## [49/50] USW00093134 (305,096 records)
## [50/50] USC00041072 (157,593 records)
##
## [OK] Saved 50 stations to: /home/mwl04747/RTricks/05_Regional_Climate_Trends/2026_7/Data/
##      File size: 16 MB
##
## Cleaning up temporary files...
##   Removing 100 .csv and .gz files
##   Freed up 498 MB of disk space
## ============================================================
## Loading Summary:
## ============================================================
##   Successfully loaded: 50 stations
##   Failed to load: 0 stations
##   Saved to RData: /home/mwl04747/RTricks/05_Regional_Climate_Trends/2026_7/Data/all_stati
##   Cleaned up temporary files: YES
## ============================================================
## Next step: process_all_stations()
```

```
# Load all CSV files and save as RData format
# cleanup = TRUE removes temporary files to save disk space
load_and_save_stations(cleanup = TRUE)
```

R Concept: RData Files
RData files (`.RData`) are R's native format for saving data. They:

- Are compressed (smaller file size)
- Load faster than CSV files
- Preserve R data types (dates, factors, etc.)
- Can store multiple objects in one file

7 Learning R: Working with Dates

One of the most important skills in data analysis is handling dates properly. Let's learn how R handles dates using our climate data.

7.1 The Problem: NOAA Date Format

NOAA stores dates as 8-digit integers. For example:

- `20230715` means July 15, 2023
- `19610101` means January 1, 1961

R doesn't automatically recognize this format. We need to convert it.

```
## Raw DATE values from NOAA:
##  [1] 18670601 18670602 18670603 18670604 18670605 18670606 18670607 18670608
##  [9] 18670609 18670610
##
## Data type: integer
```

```
# Look at the raw date format
cat("Raw DATE values from NOAA:\n")
print(head(example_station$DATE, 10))
cat("\nData type:", class(example_station$DATE), "\n")
```

7.2 Converting Dates with as.Date()

The `as.Date()` function converts text to proper date objects.

```
## Original integer: 20230715
## As character:     20230715
## As Date object:   2023-07-15
## Data type now:    Date
```

```
# Convert integer to character, then to Date
# The format string tells R how to interpret the text:
#   %Y = 4-digit year
#   %m = 2-digit month
#   %d = 2-digit day

example_date <- 20230715
date_as_char <- as.character(example_date)
date_proper <- as.Date(date_as_char, format = "%Y%m%d")

cat("Original integer:", example_date, "\n")
cat("As character:    ", date_as_char, "\n")
cat("As Date object:  ", as.character(date_proper), "\n")
```

R Concept: Date Format Codes
Common format codes for dates:

- `%Y` – 4-digit year (2023)
- `%y` – 2-digit year (23)
- `%m` – Month as number (07)
- `%d` – Day of month (15)
- `%b` – Abbreviated month name (Jul)
- `%B` – Full month name (July)

7.3 Extracting Month and Year

Once we have a Date object, we can extract components:

```
## Date: 2023-07-15
## Month: 7
## Year: 2023
## Day: 15
```

```
# Extract month and year from a date
my_date <- as.Date("2023-07-15")

# Use format() to extract parts
month_num <- as.numeric(format(my_date, "%m"))
year_num <- as.numeric(format(my_date, "%Y"))
day_num <- as.numeric(format(my_date, "%d"))

cat("Date:", as.character(my_date), "\n")
cat("Month:", month_num, "\n")
cat("Year:", year_num, "\n")
cat("Day:", day_num, "\n")
```

7.4 The fixDates.fun() Function

Our `fixDates.fun()` function does all of this for station data:

```
## New columns added:
##   Ymd (Date): Date
##   MONTH: numeric - range: 1 to 12
##   YEAR: numeric - range: 1867 to 2026
```

```
# The fixDates.fun() function adds three new columns:
#   Ymd   - proper Date object
#   MONTH - month number (1-12)
#   YEAR  - 4-digit year

station_a <- fixDates.fun(example_station)

# Check the new columns
head(station_a[, c("DATE", "Ymd", "MONTH", "YEAR")])
```

8 Learning R: Unit Conversions

NOAA stores values in scaled units to save storage space. We need to convert them.

8.1 Understanding NOAA Units

- **Temperature (TMAX, TMIN):** Stored in tenths of degrees Celsius
  - Value of 235 = 23.5C
  - Value of -50 = -5.0C
- **Precipitation (PRCP):** Stored in tenths of millimeters
  - Value of 50 = 5.0 mm

```
## Raw TMAX values (tenths of degrees C):
## [1] -6 11 17 22 -6
##
## These need to be divided by 10 to get actual temperatures.
```

```
# Look at raw values before conversion
tmax_raw <- subset(station_a, ELEMENT == "TMAX")
cat("Raw TMAX values (tenths of degrees C):\n")
print(head(tmax_raw$VALUE, 5))
```

8.2 The fixValues.fun() Function

```
## After conversion (degrees C):
## [1] -0.6  1.1  1.7  2.2 -0.6
```

```
# Apply unit conversion
station_b <- fixValues.fun(station_a)

# The function divides temperature by 10
# and precipitation by 10 to get proper units
```

9 Learning R: Data Subsetting

Subsetting is one of the most important skills in R. Let's practice with our climate data.

9.1 Using subset()

```
## TMAX records: 48982
## Summer records: 56751
## Summer TMAX records: 12388
```

```
# Subset for just TMAX data
tmax_data <- subset(station_b, ELEMENT == "TMAX")

# Subset for summer months (June, July, August)
# The %in% operator checks if values are in a list
summer_data <- subset(station_b, MONTH %in% c(6, 7, 8))

# Combine conditions with & (AND)
hot_summer <- subset(station_b,
                     ELEMENT == "TMAX" & MONTH %in% c(6, 7, 8))

# Use | for OR conditions
# Use != for "not equal"
```

R Concept: Logical Operators

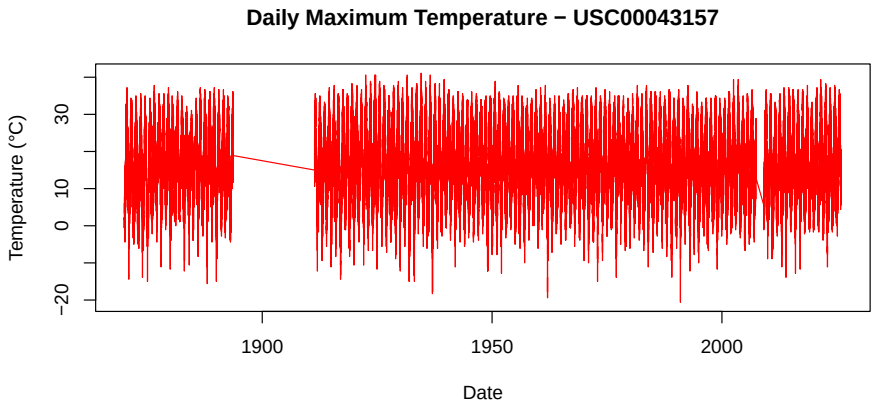
- `==` Equal to
- `!=` Not equal to

- `>`, `<` Greater/less than
- `>=`, `<=` Greater/less than or equal
- `&` AND (both conditions must be true)
- `|` OR (either condition can be true)
- `%in%` Is the value in this list?

10 Learning R: Creating Plots

Visualization is crucial for understanding climate data. Let's create some basic plots.

10.1 Basic Time Series Plot



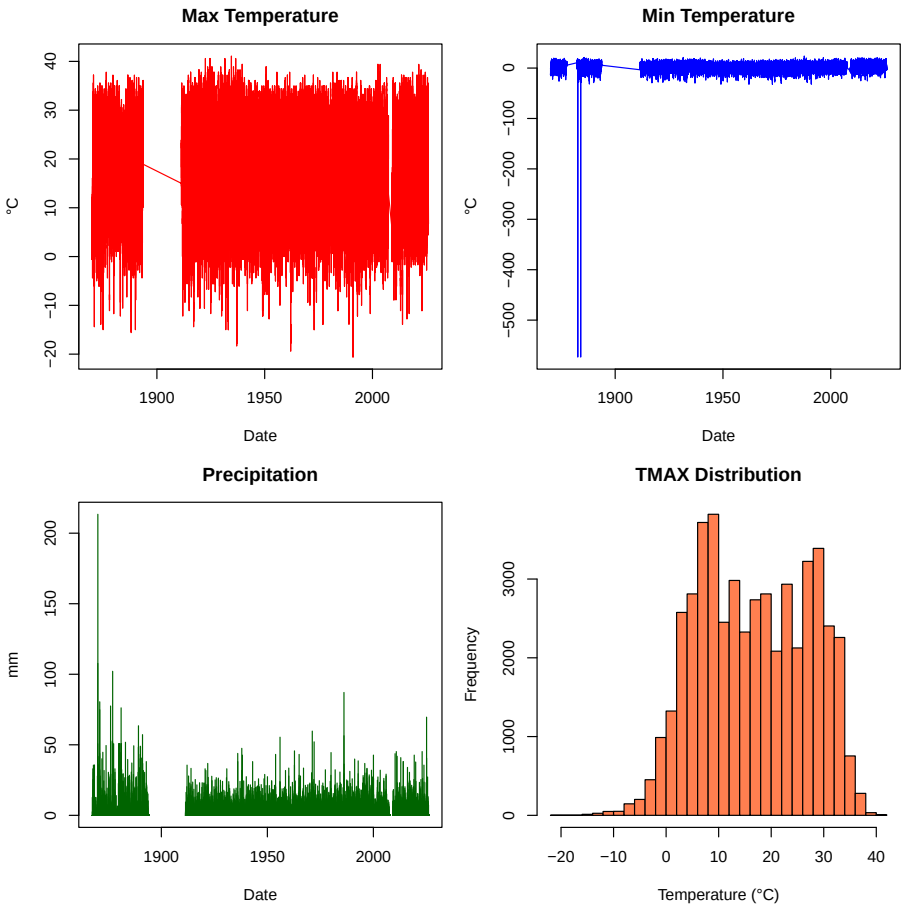

```
# Get TMAX data
tmax_data <- subset(station_b, ELEMENT == "TMAX")

# Basic plot with plot()
plot(VALUE ~ Ymd, data = tmax_data,
     type = "l",          # "l" for lines, "p" for points
     col = "red",         # color
     main = "Daily Maximum Temperature",  # title
     xlab = "Date",       # x-axis label
     ylab = "Temperature (C)")  # y-axis label
```

R Concept: The Formula Syntax

In R, `y ~ x` means "y as a function of x". This is called **formula syntax** and is used throughout R for:

- Plotting: `plot(y ~ x)`
- Regression: `lm(y ~ x)`
- Aggregation: `aggregate(y ~ group, ...)`

10.2 Multiple Panels




```
# Set up 2x2 panel of plots
par(mfrow = c(2, 2),    # 2 rows, 2 columns
```

```
    mar = c(4, 4, 3, 1)) # margins: bottom, left, top, right

# Plot 1: TMAX time series
plot(VALUE ~ Ymd, data = tmax_data, type = "l", col = "red",
     main = "Max Temperature", xlab = "Date", ylab = "C")

# Plot 2: TMIN time series
plot(VALUE ~ Ymd, data = tmin_data, type = "l", col = "blue",
     main = "Min Temperature", xlab = "Date", ylab = "C")

# Plot 3: Precipitation (type = "h" for histogram-like bars)
plot(VALUE ~ Ymd, data = prcp_data, type = "h", col = "green",
     main = "Precipitation", xlab = "Date", ylab = "mm")

# Plot 4: Histogram
hist(tmax_data$VALUE, breaks = 30, col = "coral",
     main = "TMAX Distribution", xlab = "Temperature (C)")

par(mfrow = c(1, 1))  # Reset to single panel
```

11 Learning R: Data Aggregation

Aggregation means summarizing data by groups. For climate analysis, we aggregate daily data to monthly values.

11.1 Using aggregate()

```
## Monthly TMAX data (first 12 rows):
##    MONTH YEAR      TMAX
## 1      1 1870  4.506452
## 2      2 1870  7.021429
## 3      3 1870  7.141935
## 4      4 1870 16.060000
## 5      5 1870 20.641935
## 6      6 1870 25.926667
## 7      7 1870 31.161290
## 8      8 1870 29.464516
## 9      9 1870 24.350000
## 10    10 1870 16.787097
## 11    11 1870 10.700000
## 12    12 1870 -1.319355
```

```
# Calculate monthly mean TMAX
tmax_data <- subset(station_b, ELEMENT == "TMAX")

# aggregate() syntax:
#   VALUE ~ MONTH + YEAR means:
#     "aggregate VALUE by MONTH and YEAR"
#   FUN = mean means: calculate the mean
#   na.rm = TRUE means: ignore missing values

monthly_tmax <- aggregate(VALUE ~ MONTH + YEAR,
                          data = tmax_data,
                          FUN = mean,    # or sum, median, sd, etc.
                          na.rm = TRUE)

# Rename the result column
names(monthly_tmax)[3] <- "TMAX"
```

R Concept: Aggregation Functions
Common functions to use with `aggregate()`:

- `mean` – Average
- `sum` – Total
- `median` – Middle value
- `sd` – Standard deviation
- `min`, `max` – Extremes
- `length` – Count of observations

12 Understanding Climate Anomalies

12.1 What is an Anomaly?

An **anomaly** is the difference between an observed value and a reference value (the "normal"):

$$Anomaly = Observed - Normal$$

- Positive anomaly = warmer/wetter than normal
- Negative anomaly = cooler/drier than normal

12.2 Climate Normals (1961-1990)

The standard reference period is 1961-1990, called the "climate normal." We calculate the average for each month during this period.

```
## Climate Normals (1961-1990) for TMAX:
##    MONTH    NORMAL
## 1      1  4.380645
## 2      2  7.723009
## 3      3 11.184409
## 4      4 15.546042
## 5      5 20.561123
## 6      6 25.246000
## 7      7 29.835699
## 8      8 29.369032
## 9      9 25.148562
## 10    10 19.123763
## 11    11  9.909540
## 12    12  5.003656
```

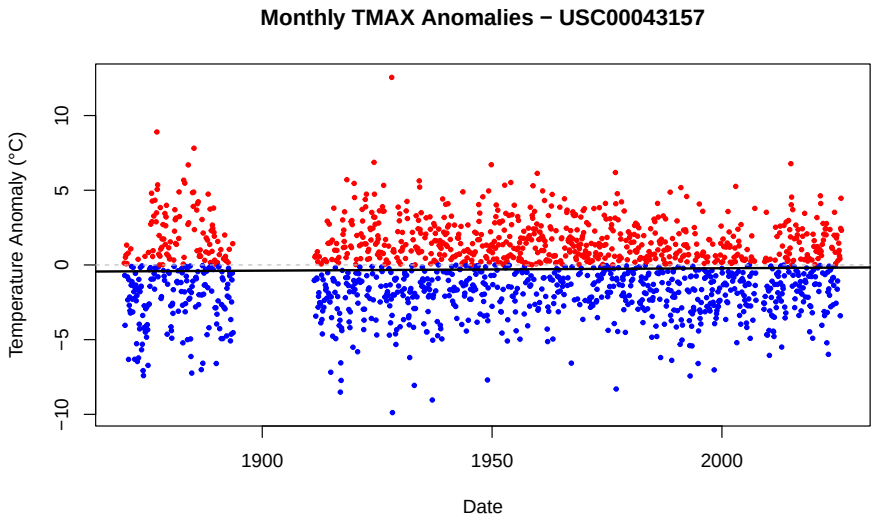
```
# Subset data to the normal period (1961-1990)
normal_period <- subset(station_b,
                        Ymd >= as.Date("1961-01-01") &
                        Ymd <= as.Date("1990-12-31") &
                        ELEMENT == "TMAX")

# Calculate normals: mean TMAX for each month
tmax_normals <- aggregate(VALUE ~ MONTH,
                          data = normal_period,
                          FUN = mean, na.rm = TRUE)
names(tmax_normals) <- c("MONTH", "NORMAL")
```

12.3 Calculating Anomalies

```
## Sample of anomaly data:
##     MONTH YEAR      TMAX    NORMAL     ANOMALY
## 1       1 1870  4.506452  4.380645  0.12580645
## 270     2 1870  7.021429  7.723009 -0.70158028
## 369     3 1870  7.141935 11.184409 -4.04247312
## 492     4 1870 16.060000 15.546042  0.51395764
## 627     5 1870 20.641935 20.561123  0.08081237
## 749     6 1870 25.926667 25.246000  0.68066667
```

```
# Merge monthly values with normals by MONTH
monthly_with_normals <- merge(monthly_tmax, tmax_normals,
                              by = "MONTH")

# Calculate anomaly = observed - normal
monthly_with_normals$ANOMALY <- monthly_with_normals$TMAX -
                                  monthly_with_normals$NORMAL
```

12.4 Visualizing Anomalies




```
# Plot anomalies with colors based on sign
plot(ANOMALY ~ Date, data = monthly_with_normals,
     pch = 19, cex = 0.5,
     col = ifelse(ANOMALY > 0, "red", "blue"),  # red=warm, blue=cool
     main = "Monthly TMAX Anomalies",
     xlab = "Date", ylab = "Temperature Anomaly (C)")

# Add reference line at zero
abline(h = 0, lty = 2, col = "gray")

# Add trend line using linear regression
```

```
trend <- lm(ANOMALY ~ Date, data = monthly_with_normals)
abline(trend, col = "black", lwd = 2)
```

13 Process All Stations

Now we apply these techniques to all 50 stations.

```
## Loading station data from RData file...
## ============================================================
##   Processing All Stations for Spatial Analysis
## ============================================================
## Processing 50 stations...
##
## [1/50] USC00043157 ... OK
## [2/50] USW00023271 ... OK
## [3/50] USC00042294 ... OK
## [4/50] USC00046074 ... OK
## [5/50] USC00046136 ... OK
## [6/50] USC00046719 ... OK
## [7/50] USC00046826 ... OK
## [8/50] USC00047821 ... OK
## [9/50] USC00047902 ... OK
## [10/50] USC00049866 ... OK
## [11/50] USC00040693 ... OK
## [12/50] USC00044412 ... OK
## [13/50] USC00048351 ... OK
## [14/50] USC00041614 ... OK
## [15/50] USC00046730 ... OK
## [16/50] USC00047880 ... OK
## [17/50] USC00044259 ... OK
## [18/50] USC00049073 ... OK
## [19/50] USC00043161 ... OK
## [20/50] USC00047195 ... OK
## [21/50] USC00048702 ... OK
## [22/50] USW00023157 ... OK
## [23/50] USC00040943 ... OK
## [24/50] USC00043875 ... OK
## [25/50] USC00042805 ... OK
## [26/50] USC00043191 ... OK
## [27/50] USC00048839 ... OK
## [28/50] USC00049367 ... OK
## [29/50] USC00043747 ... OK
## [30/50] USC00044890 ... OK
```

```
## [31/50] USC00042239 ... OK
## [32/50] USC00045532 ... OK
## [33/50] USC00049452 ... OK
## [34/50] USC00041018 ... OK
## [35/50] USC00044223 ... OK
## [36/50] USC00044500 ... OK
## [37/50] USC00047965 ... OK
## [38/50] USC00046168 ... OK
## [39/50] USC00044997 ... OK
## [40/50] USC00046508 ... OK
## [41/50] USC00046506 ... OK
## [42/50] USC00046252 ... OK
## [43/50] USC00040383 ... OK
## [44/50] USC00041912 ... OK
## [45/50] USC00046399 ... OK
## [46/50] USC00049855 ... OK
## [47/50] USC00048353 ... OK
## [48/50] USC00049699 ... OK
## [49/50] USW00093134 ... OK
## [50/50] USC00041072 ... OK
## ============================================================
## Processing Summary:
## ============================================================
##   Successfully processed: 50 stations
##   Time elapsed: 0.7 minutes
## ============================================================
## [OK] Saved spatial trends data
```

```
# Process all stations at once
# This function:
#   1. Fixes dates for each station
#   2. Converts units
#   3. Calculates monthly values
#   4. Calculates normals
#   5. Calculates anomalies
#   6. Fits trend lines
#   7. Saves results

all_station_trends <- process_all_stations_for_spatial(verbose = TRUE)
```

14 Create Spatial Objects

Finally, we convert our trend data to spatial format for mapping in Lab 3.

```
## [OK] Created spatial objects:
##      trends_sf: 50 features
##      trends_sp: 50 features
```

```
# Create spatial objects for mapping
# This creates:
#   trends_sf - Simple Features format (modern)
#   trends_sp - Spatial Points format (legacy, for kriging)

spatial_objects <- create_spatial_objects(all_station_trends)
```

15 Summary Statistics

```
## ============================================================
## CLIMATE TRENDS SUMMARY FOR CA
## ============================================================
##
## Stations analyzed: 50
##
## Mean Annual Trends:
##   TMAX: 0.56 C/century
##   TMIN: 1.49 C/century
##   PRCP: -3.3 mm/century
## ============================================================
```

16 Lab 1 Summary: R Concepts Learned

In this lab, you learned:

1. **Date handling**
   - `as.Date()` converts text to dates
   - Format codes like `%Y%m%d`
   - Extracting parts with `format()`
2. **Data subsetting**

- `subset()` for filtering rows
- Logical operators (`==`, `&`, `|`, `%in%`)
- Square brackets for rows and columns

3. **Data aggregation**
   - `aggregate()` for group summaries
   - Formula syntax (`y ~ x`)
   - Common functions (mean, sum, etc.)
4. **Basic plotting**
   - `plot()` for scatter/line plots
   - `hist()` for histograms
   - `par()` for multiple panels
   - Adding elements with `abline()`
5. **Climate concepts**
   - Climate normals (1961-1990 baseline)
   - Anomalies (deviation from normal)
   - Trends (change over time)

17 Next Steps

You're ready for Lab 2, where you will:

- Analyze monthly and seasonal trend patterns
- Learn statistical significance testing
- Compare warming patterns across your state
- Create more advanced visualizations