Timeseries and Autocorrelation

Willem Vervoort, Michaela Dolk & Floris van Ogtrop

2020-01-30

```
# root dir
knitr::opts_knit$set(root.dir = "D:/cloudstor/Virtual Experiments/VirtExp")
knitr::opts_chunk$set(echo = TRUE)
# LOAD REQUIRED PACKAGES # #####
library(pander)
library(tidyverse)
library(xts)
library(zoo)
library(deseasonalize)
```

This rmarkdown document and the resulting pdf are stored on github. All directories (apart from the root working directory) refer to the directories in this repository

Introduction

This document is related to the manuscript "Disentangling climate change trends in Australian streamflow" (vervoort et al.), submitted. This is a document to respond to some questions from Tim Peterson (Monash University) in relation to the significance of the trends in the temperature data. In particular the question was about the # The data
Using the datasets that were developed earlier, we can load in the daily data for streamflow, rainfall and temperature. The difference is that the Mann Kendal analysis should concentrate on analysing the anomalies rather than the actual data, and therefore we have to resummarise to weekly data after calculating the anomalies.

A further point to note is that the observed data contains missing values, which remain in the analysis, but dissapear in the summarised data. The gridded rainfall data does not contain missing data as this is interpolated predicted data.

```
load("data/DailyDataIncludingGridded.Rdata")
load("data/ClimCh_project_MD.Rdata")
```

Deseasonalise the data

The next step is to calculate the anomalies by deseasonalising the original daily data using the package `deseasonalize` in R: deseasonalize.

```
# daily observed flow
flow_deseas <- flow_zoo
# now assign to a new dataframe
for (i in (seq_along(Stations[,1]))) {
  foo <- flow_zoo[,i]
# replace missing values with mean flow
  bad <- is.na(foo)
  foo[bad] <- mean(foo,na.rm=T)
```

```
  flow_deseas[,i]  <- ds(as.ts(foo),ic="AIC")$z
# put NA values back
  flow_deseas[bad,i] <- NA
}
# daily observed rainfall
rain_deseas <- rain_zoo
# now assign to a new dataframe
for (i in seq_along(Stations[,1])) {
  foo <- rain_zoo[,i]
  # replace NA values with mean flow
  bad <- is.na(foo)
  foo[bad] <- mean(foo,na.rm=T)
  rain_deseas[,i]  <- ds(as.numeric(foo),ic="AIC")$z
  # put NA values back
  rain_deseas[bad,i] <- NA
}

# daily observed maximum temperature
maxT_deseas <- maxT_zoo
# now assign to a new dataframe
for (i in seq_along(Stations[,1])) {
  foo <- maxT_zoo[,i]
  # replace NA values with mean flow
  bad <- is.na(foo)
  foo[bad] <- mean(foo,na.rm=T)
  maxT_deseas[,i]  <- ds(as.numeric(foo),ic="AIC")$z
  # put NA values back
  maxT_deseas[bad,i] <- NA
}

# do the same for the gridded rainfall data
rain_griddeseas <- rain_zoo
#
for (i in seq_along(Stations[,1])) {
  foo <- GridRainAllDataout %>%
    filter(grepl(Stations[i,1],Station)) %>%
    select(gridRain)
  foo.z <- zoo(foo,  order.by=time(rain_zoo))
  rain_griddeseas[,i]  <- ds(foo.z,ic="AIC")$z
}
```

Summarise to weekly, monthly and annual data

Similar to the original data, the package `xts` can be used to summarise the data to weekly mean values. For the flow and rainfall data, we calculate the sum by week, month and year, while for the temperature we calculate the mean.

In addition, the monthly and annual anomalies are also generated so the difference between the trend analysis on a weekly, monthly and annual scale can be checked.

```
# flow (sum flow)
flow_xts <- xts(flow_deseas,
                order.by=time(flow_deseas),
```

```
                frequency=1)

# weekly data summarising (destroys xts object)
flow_weekly <- apply(flow_xts,2,
                             function(x) apply.weekly(x,
                                                      sum,na.rm=T))
# define weekly dates
Dates <- time(apply.weekly(flow_xts[,1],sum))
# restore the xts object
flow_weekly_xts <- as.xts(flow_weekly,
                              order.by=Dates)

# rainfall (sum rainfall)
rainfall_xts <- xts(rain_deseas,
                    order.by=time(rain_deseas),
                    frequency=1)
# weekly data summarising (destroys xts object)
rainfall_weekly <- apply(rainfall_xts,2,
                             function(x) apply.weekly(x,
                                                      sum,na.rm=T))
# define weekly dates
Dates <- time(apply.weekly(rainfall_xts[,1],sum))
# restore the xts object
rainfall_weekly_xts <- as.xts(rainfall_weekly,
                              order.by=Dates)

# gridded rainfall (sum rainfall)
rainfall_grdxts <- xts(rain_griddeseas,
                    order.by=time(rain_griddeseas),
                    frequency=1)
# weekly data summarising (destroys xts object)
rainfall_grdweekly <- apply(rainfall_grdxts,2,
                             function(x) apply.weekly(x,
                                                      sum,na.rm=T))
# define weekly dates
Dates <- time(apply.weekly(rainfall_grdxts[,1],sum))
# restore the xts object
rainfall_grdweekly_xts <- as.xts(rainfall_grdweekly,
                              order.by=Dates)

# maxT
maxT_xts <- xts(maxT_deseas,
                order.by=time(maxT_deseas),
                frequency=1)
# weekly data summarising (destroys xts object)
maxT_weekly <- apply(maxT_xts,2,
                             function(x) apply.weekly(x,
                                                      sum,na.rm=T))
# define weekly dates
Dates <- time(apply.weekly(maxT_xts[,1],sum))
# restore the xts object
maxT_weekly_xts <- as.xts(maxT_weekly,
                              order.by=Dates)
```

```
# Also calculate monthly and annual for MaxT
# monthly data summarising (destroys xts object)
maxT_monthly <- apply(maxT_xts,2,
                              function(x) apply.monthly(x,
                                                        sum,na.rm=T))
# define monthly dates
Dates <- time(apply.monthly(maxT_xts[,1],sum))
# restore the xts object
maxT_monthly_xts <- as.xts(maxT_monthly,
                               order.by=Dates)
# annual data (destroys xts object)
maxT_annual <- apply(maxT_xts,2,
                              function(x) apply.yearly(x,
                                                       sum,na.rm=T))
# define annual dates
Dates <- time(apply.yearly(maxT_xts[,1],sum))
# restore the xts object
maxT_annual_xts <- as.xts(maxT_annual,
                               order.by=Dates)
```

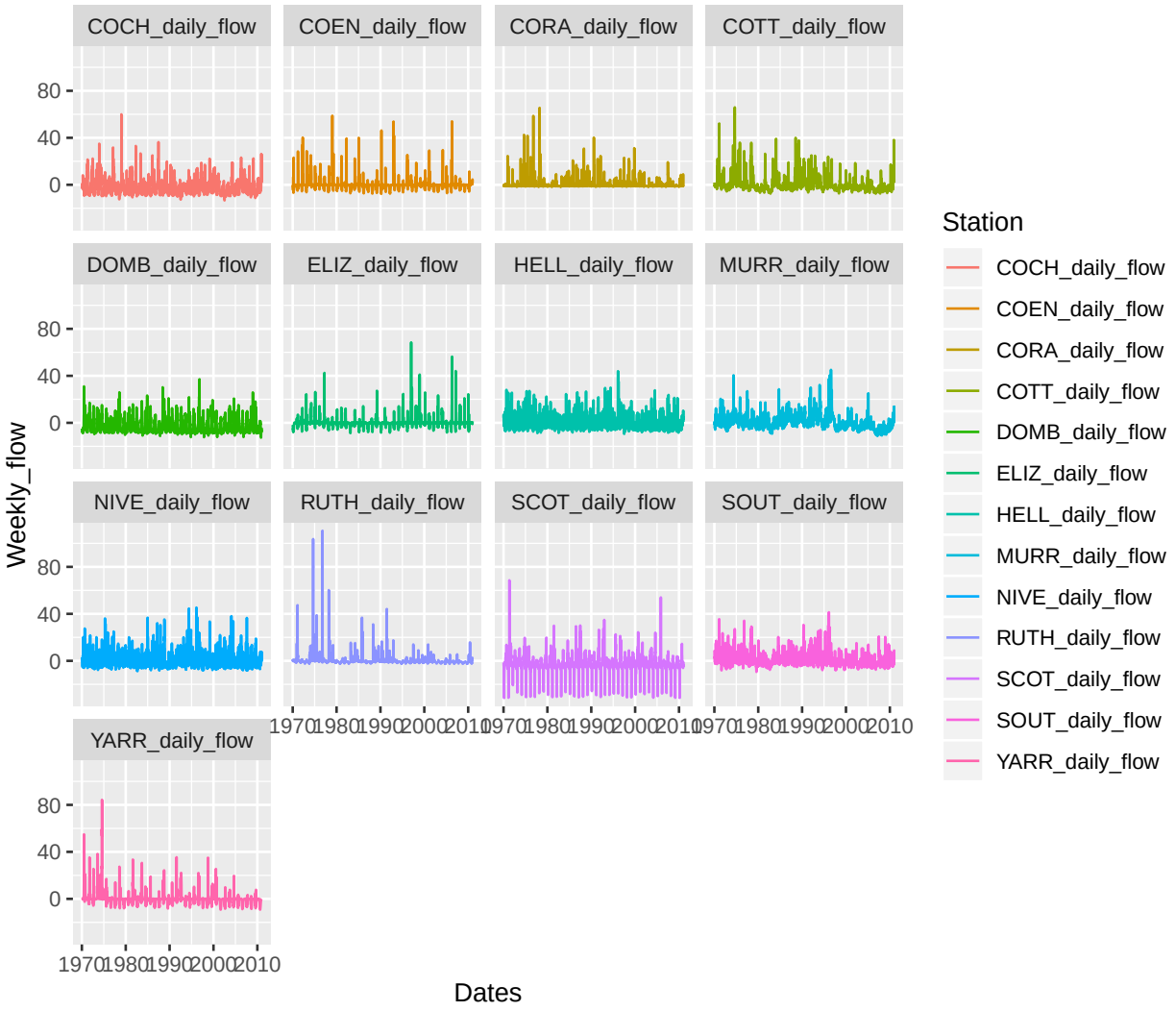
plot the residual time series and the autocorrelation of the time series

Using `ggplot()` we can quite easily plot the time series. For the autocorrelation we first need to calculate the acf data frame and the pacf data frame before we can plot.

residual time series

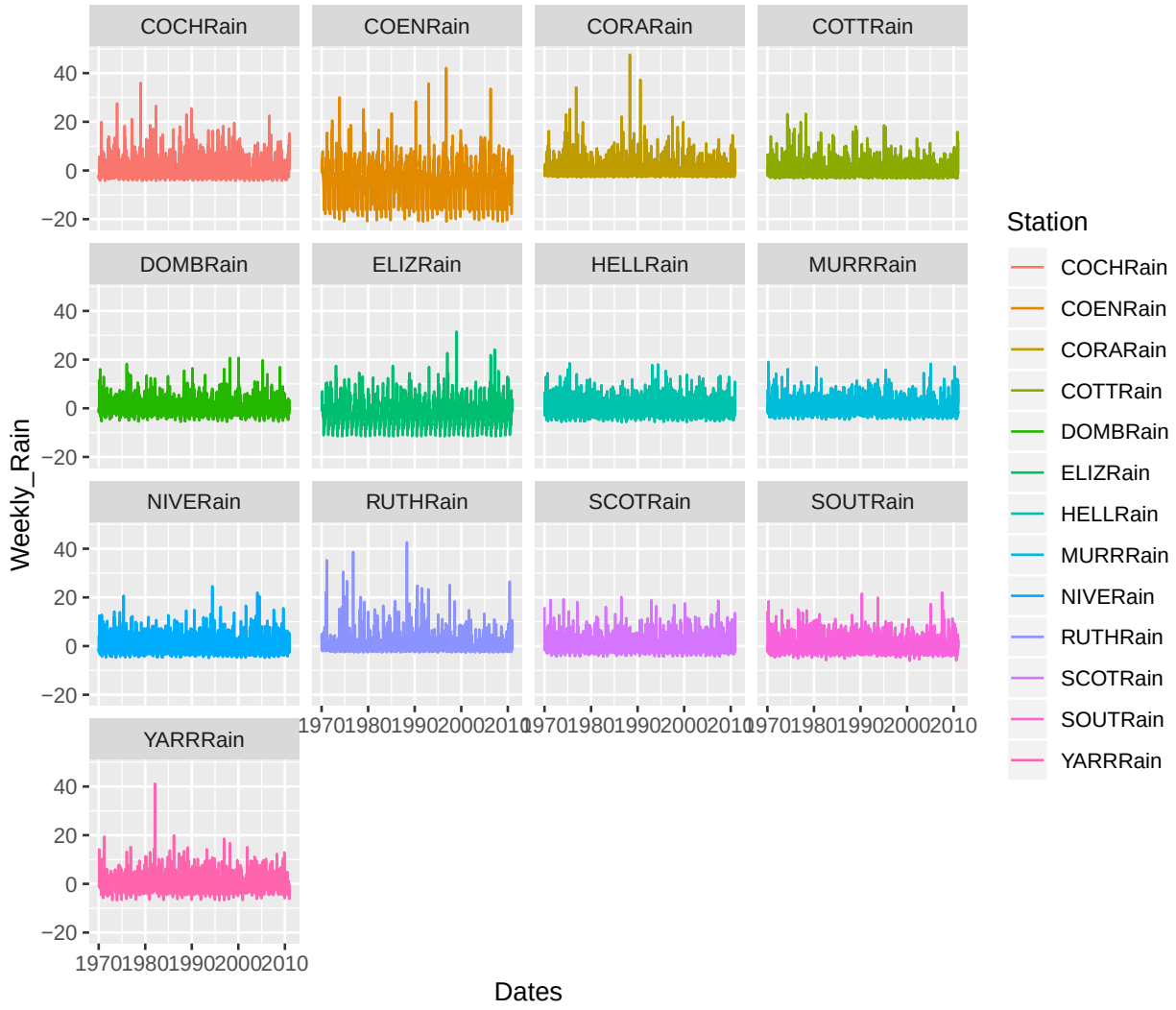
Flow

```
Dates <- time(apply.weekly(flow_xts[,1],sum))
as_tibble(flow_weekly) %>%
  mutate(Dates=Dates) %>%
  gather(key="Station", value="Weekly_flow", COTT_daily_flow:DOMB_daily_flow) %>%
  ggplot(aes(Dates,Weekly_flow, col=Station)) + geom_line() + facet_wrap(~Station)
```



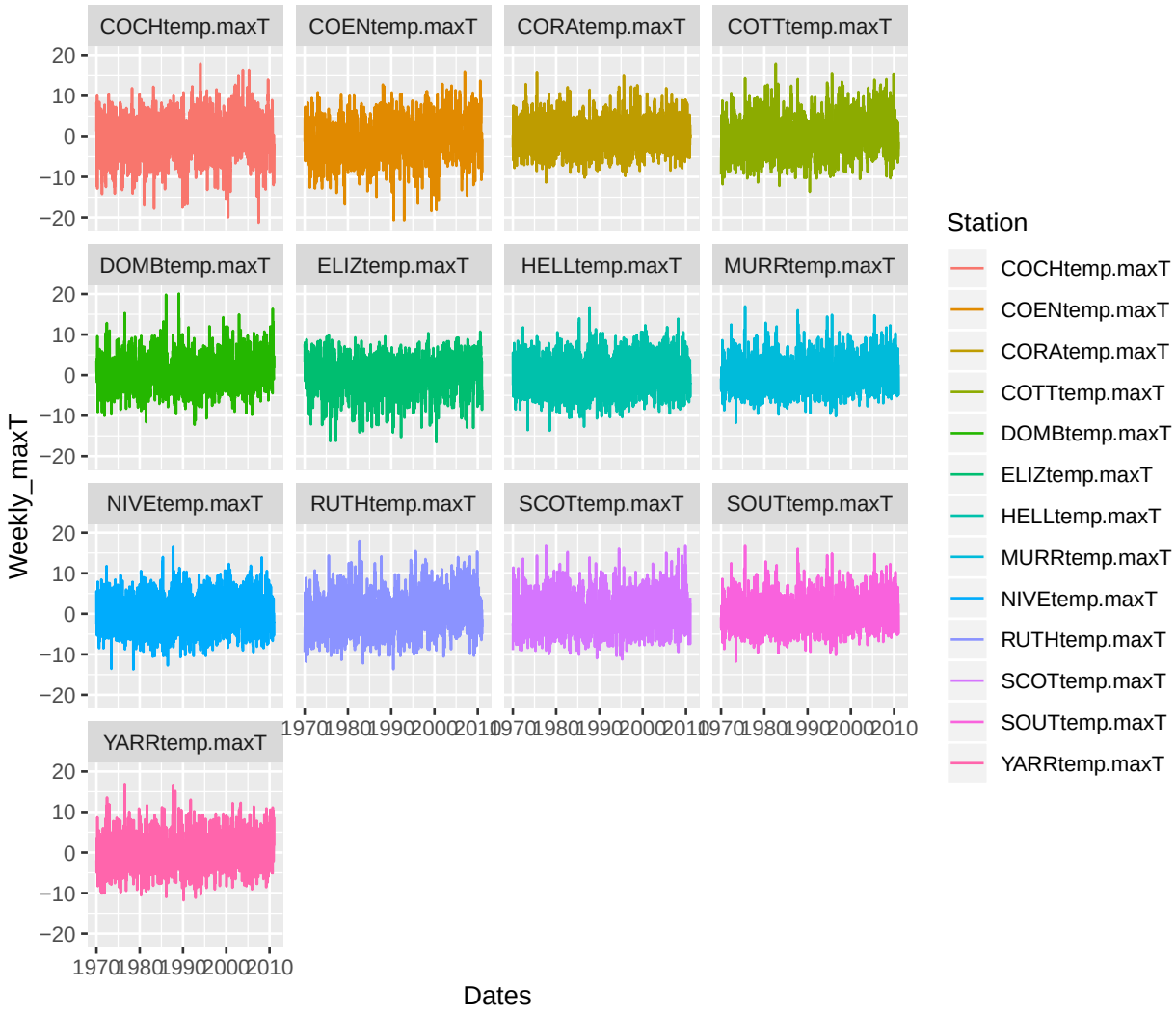
Gridded Rainfall

```
Dates <- time(apply.weekly(rainfall_grdxts[,1],sum))
as_tibble(rainfall_grdweekly) %>%
  mutate(Dates=Dates) %>%
  gather(key="Station", value="Weekly_Rain", COTTRain:DOMBRain) %>%
  ggplot(aes(Dates,Weekly_Rain, col=Station)) + geom_line() + facet_wrap(~Station)
```



Maximum Temperature

```
Dates <- time(apply.weekly(maxT_xts[,1],sum))
as_tibble(maxT_weekly) %>%
  mutate(Dates=Dates) %>%
  gather(key="Station", value="Weekly_maxT", COTTtemp.maxT:DOMBtemp.maxT) %>%
  ggplot(aes(Dates,Weekly_maxT, col=Station)) + geom_line() + facet_wrap(~Station)
```



Clear trend in the some of the timeseries

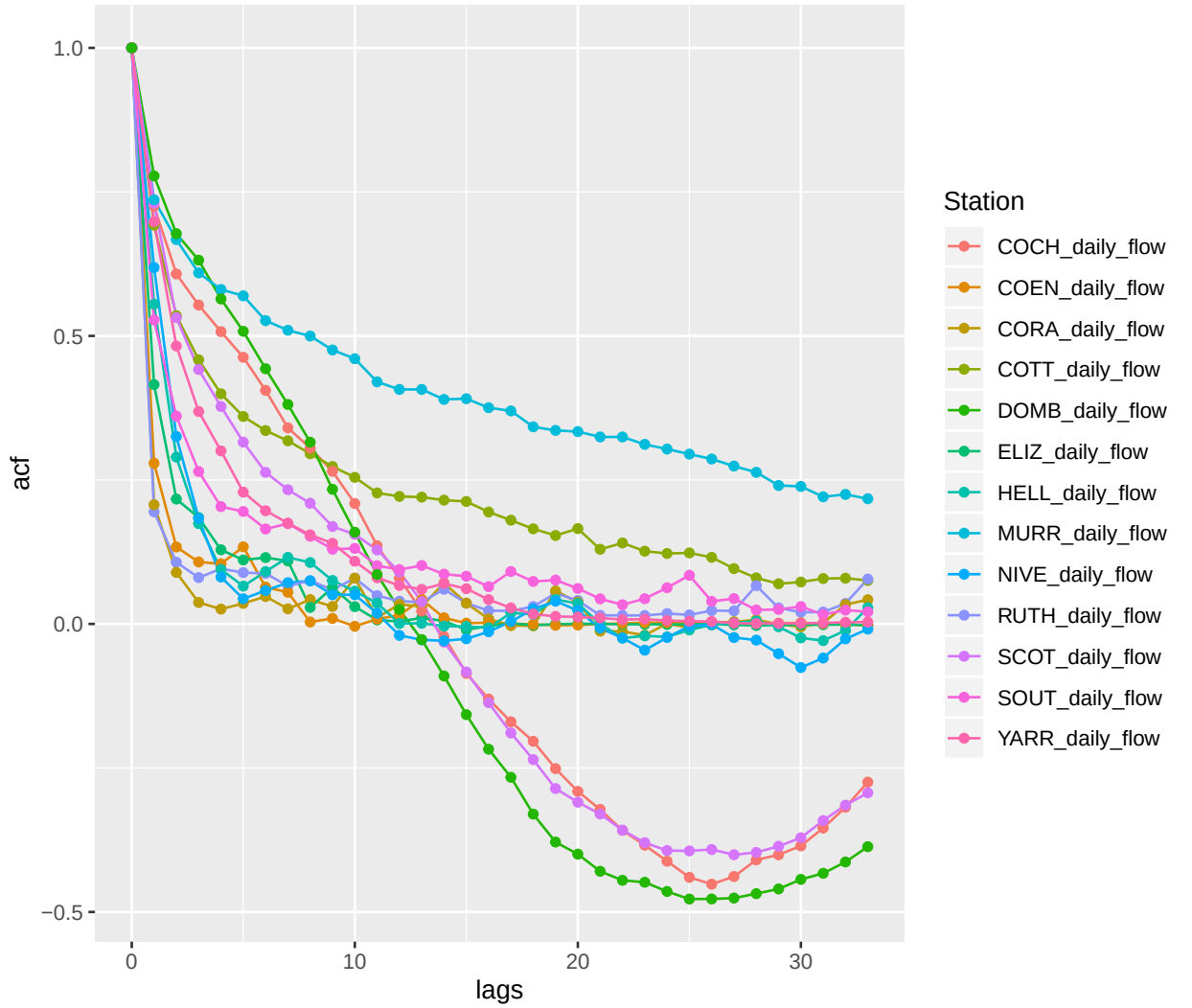
acf

First calculate the `acf()` for each series and then plot them

Flow

```
acf_flow <- as_tibble(flow_weekly) %>%
  map_df(~acf(.,plot=F)$acf)

# add the lags and plot
acf_flow %>%
  mutate(lags = acf(flow_weekly[,1], plot=F)$lag) %>%
  gather(key="Station", value="acf",COTT_daily_flow:DOMB_daily_flow) %>%
  ggplot(aes(lags,acf,col=Station)) + geom_line() + geom_point()
```

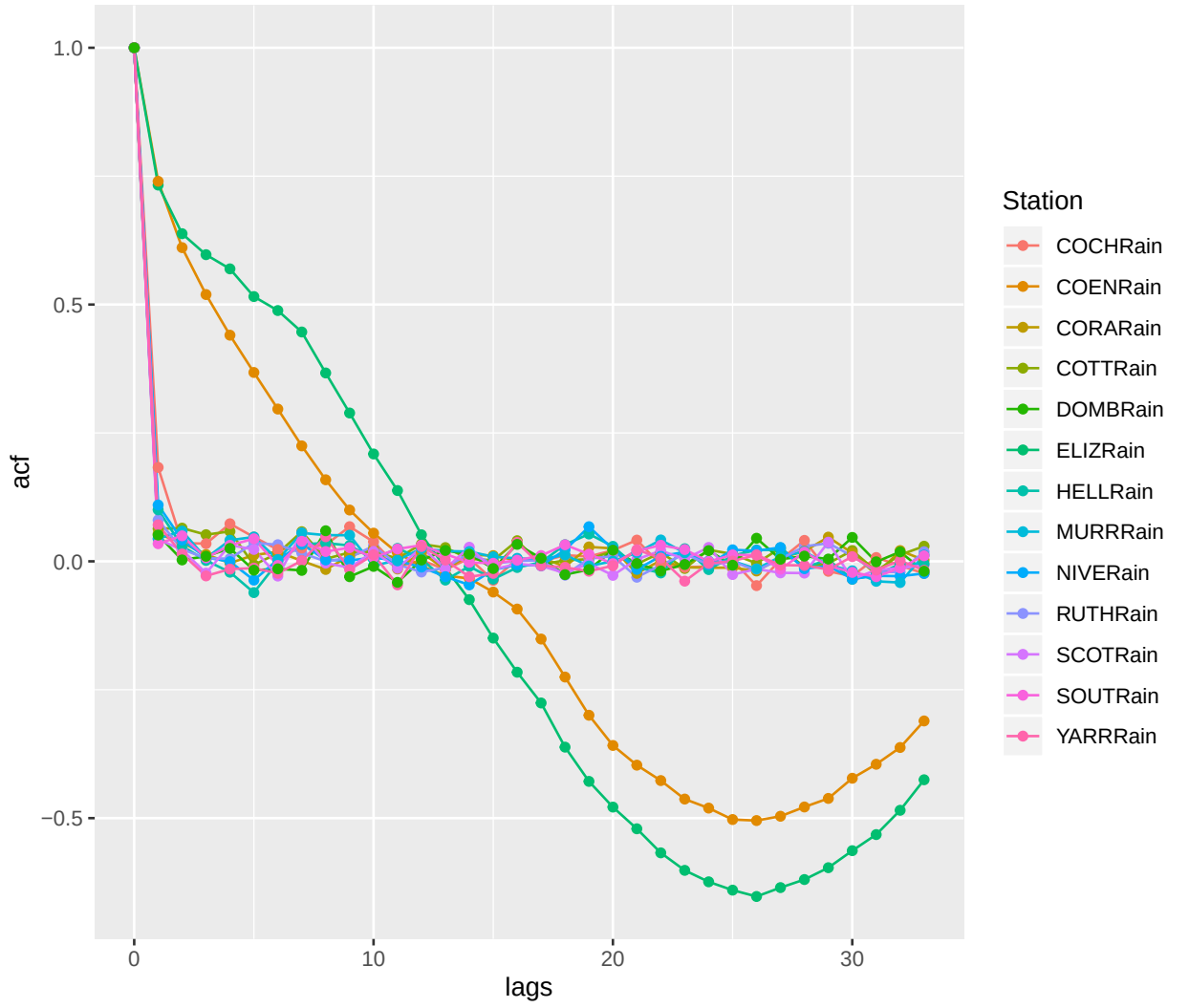


Clearly there is some signigficant autocorrelation in the residuals of the flow data for some of the stations, in particular MURR, COTT, DOMB, SCOT and COCH. This might invalidate the AR(1) assumption in the LTP Mann Kendall (Hamed, 2008)

Rainfall

```
acf_rain <- as_tibble(rainfall_grdweekly) %>%
  map_df(~acf(.,plot=F)$acf)

# add the lags and plot
acf_rain %>%
  mutate(lags = acf(rainfall_grdweekly[,1], plot=F)$lag) %>%
  gather(key="Station", value="acf",COTTRain:DOMBRain) %>%
  ggplot(aes(lags,acf,col=Station)) + geom_line() + geom_point()
```

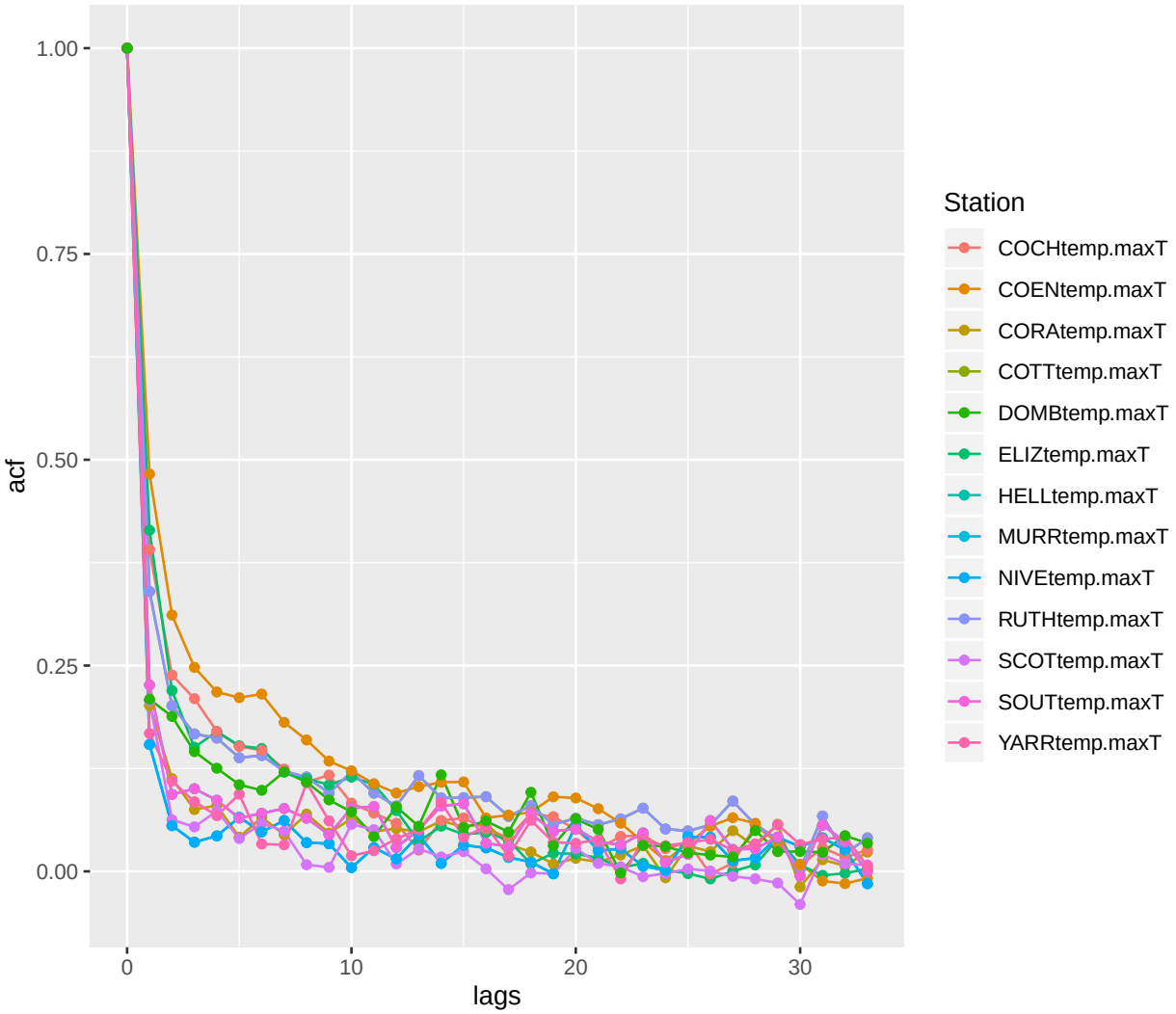


Only two of the gridded rainfall resiual data sets show significant autocorrelation. ELIZ and COEN (Northern Australia)

Maximum Temperature

```
acf_maxT <- as_tibble(maxT_weekly) %>%
  map_df(~acf(.,plot=F)$acf)

# add the lags and plot
acf_maxT %>%
  mutate(lags = acf(maxT_weekly[,1], plot=F)$lag) %>%
  gather(key="Station", value="acf",COTTtemp.maxT:DOMBtemp.maxT) %>%
  ggplot(aes(lags,acf,col=Station)) + geom_line() + geom_point()
```



Overall autocorrelation is low, except for maybe COEN (Northern Australia)

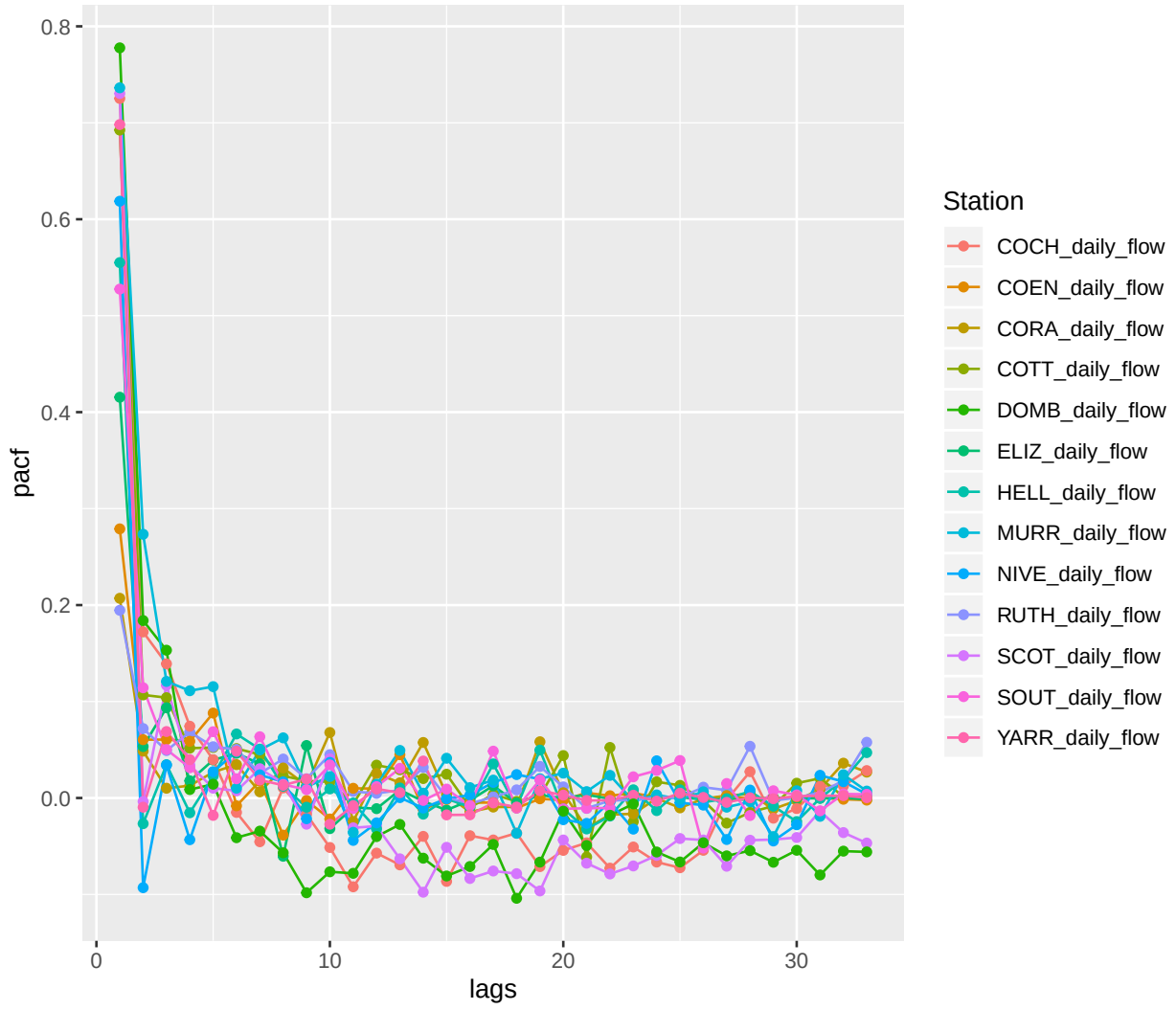
PACF

However, from a timeseries modelling perspective, it might be better to look at the partial autocorrelation function rather than the autocorrelation function. As the partial autocorrelation function takes out the effect of the preceding lags, and is better for identifying the order of the timeseries model that can be used to model the timeseries.

Flow

```
pacf_flow <- as_tibble(flow_weekly) %>%
  map_df(~pacf(.,plot=F)$acf)

# add the lags and plot
pacf_flow %>%
  mutate(lags = pacf(flow_weekly[,1], plot=F)$lag) %>%
  gather(key="Station", value="pacf",COTT_daily_flow:DOMB_daily_flow) %>%
  ggplot(aes(lags,pacf,col=Station)) + geom_line() + geom_point()
```

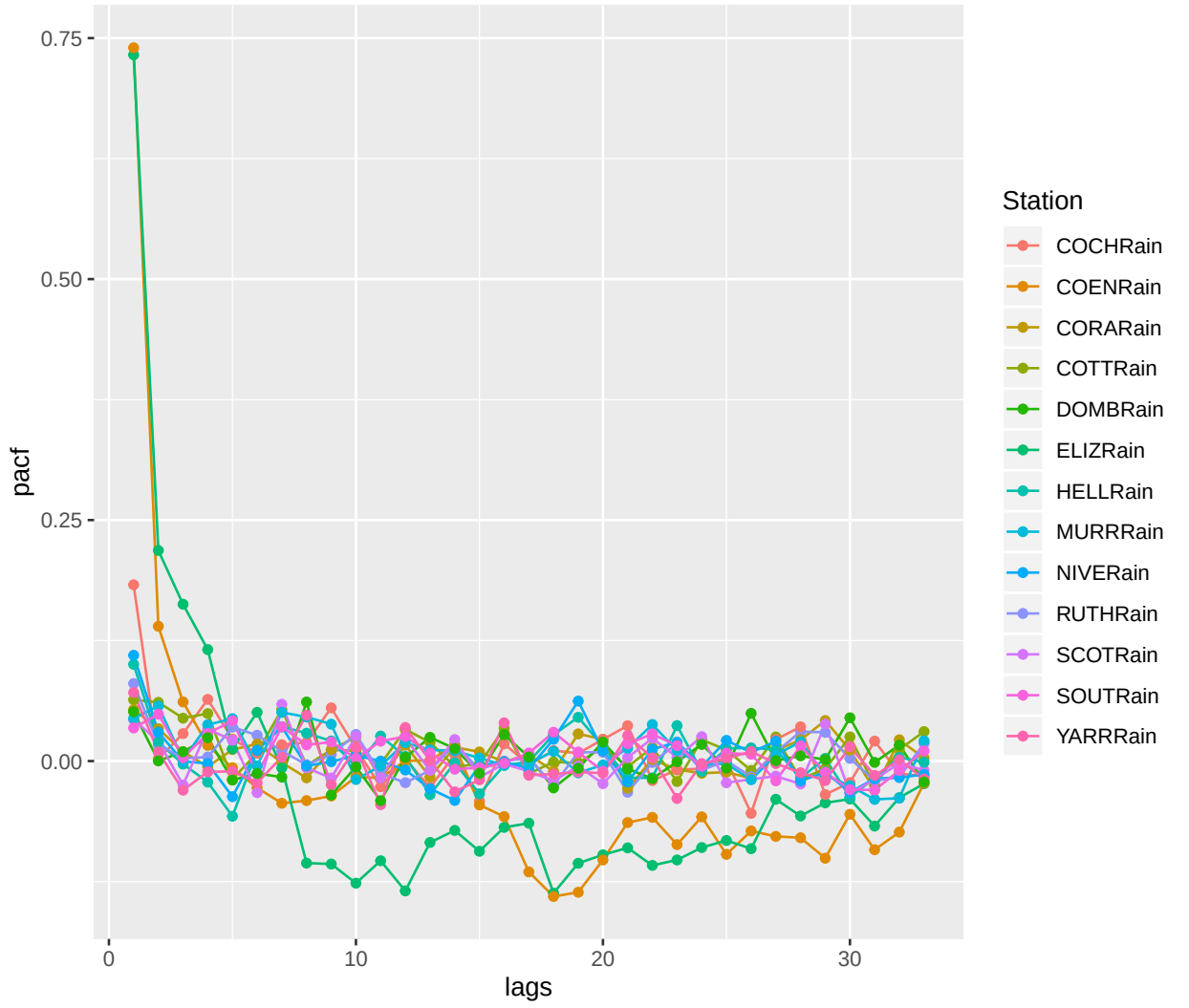


This removes most of the observed autocorrelation and leaves only the lag1 as a significant spike: AR1 is the correct model.

Rainfall

```
pacf_rain <- as_tibble(rainfall_grdweekly) %>%
  map_df(~pacf(.,plot=F)$acf)

# add the lags and plot
pacf_rain %>%
  mutate(lags = pacf(rainfall_grdweekly[,1], plot=F)$lag) %>%
  gather(key="Station", value="pacf",COTTRain:DOMBRain) %>%
  ggplot(aes(lags,pacf,col=Station)) + geom_line() + geom_point()
```

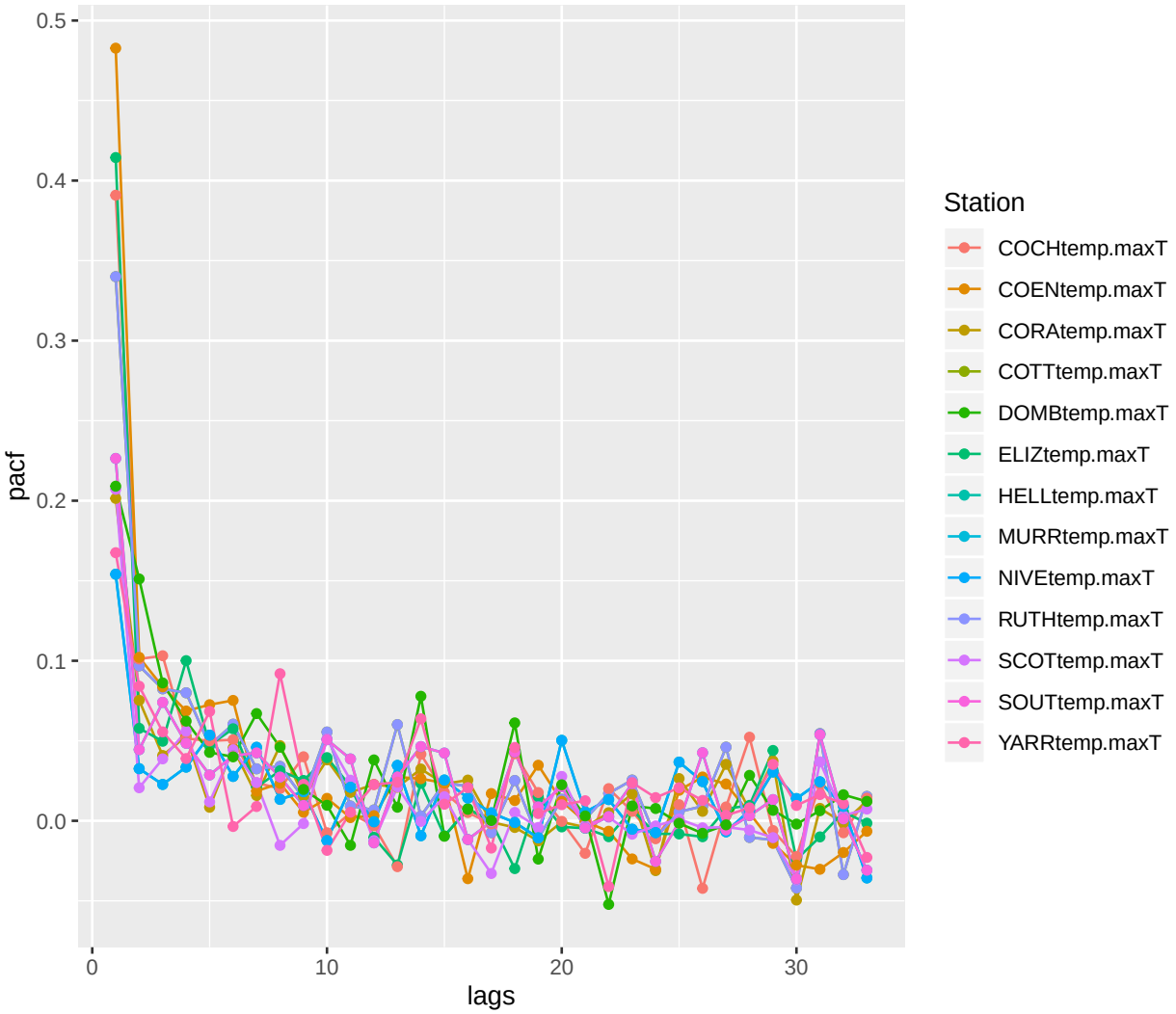


Only two of the gridded rainfall resiual data sets show some autocorrelation past lag 1. ELIZ and COEN (Northern Australia). Most rainfall data seems totally uncorrelated from a modelling perspective.

Maximum Temperature

```
pacf_maxT <- as_tibble(maxT_weekly) %>%
  map_df(~pacf(.,plot=F)$acf)

# add the lags and plot
pacf_maxT %>%
  mutate(lags = pacf(maxT_weekly[,1], plot=F)$lag) %>%
  gather(key="Station", value="pacf",COTTtemp.maxT:DOMBtemp.maxT) %>%
  ggplot(aes(lags,pacf,col=Station)) + geom_line() + geom_point()
```



Maximum temperature has a significant spike at lag 1 in the PACF, but beyond this there is very little: AR1 is the correct model.