Chapter 19
Big Longitudinal Data Analysis

The time-varying (longitudinal) characteristics of large information flows represent a special case of the complexity and the dynamic multi-scale nature of big biomedical data that we discussed in the DSPA Motivation section. Previously, in Chap. 4, we saw space-time (4D) functional magnetic resonance imaging (fMRI) data, and in Chap. 16 we discussed streaming data, which also has a natural temporal dimension. Now we will go deeper into managing, modeling and analyzing big longitudinal data.

In this Chapter, we will expand our predictive data analytic strategies specifically for analyzing big longitudinal data. We will interrogate datasets that track the same type of information, for the same subjects, units or locations, over a period of time. Specifically, we will present time series analysis, forecasting using autoregressive integrated moving average (ARIMA) models, structural equation models (SEM), and longitudinal data analysis via linear mixed models.

19.1 Time Series Analysis

Time series analysis relies on models like ARIMA (Autoregressive integrated moving average) that utilize past longitudinal information to predict near future outcomes. Times series data tent to track univariate, sometimes multivariate, processes over a continuous time interval. The stock market, e.g., daily closing value of the Dow Jones Industrial Average index, electroencephalography (EEG) data, and functional magnetic resonance imaging provide examples of such longitudinal datasets (timeserties).

The basic concepts in time series analysis include:

- The characteristics of *(second-order) stationary time series* (e.g., first two moments are stable over time) do not depend on the time at which the series process is observed.
- *Differencing* – a transformation applied to time-series data to make it stationary. Differences between consecutive time-observations may be computed by y_t'

I. D. Dinov, *Data Science and Predictive Analytics*,
https://doi.org/10.1007/978-3-319-72347-1_19

$= y_t - y_{t-1}$. Differencing removes the level changes in the time series, eliminates trend, reduces seasonality, and stabilizes the mean of the time series. Differencing the time series repeatedly may yield a stationary time series. For example, a second order differencing:

$$\begin{aligned} y_t'' &= y_t' - y_{t-1}' \\ &= (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) . \\ &= y_t - 2y_{t-1} + y_{t-2} \end{aligned}$$

- *Seasonal differencing* is computed as a difference between one observation and its corresponding observation in the previous epoch, or season (e.g., annually, there are $m = 4$ seasons), like in this example:

$$y_t''' = y_t - y_{t-m} \quad \text{where } m = \text{number of seasons.}$$

- The differenced data may then be used to estimate an ARMA model.

We will use the Beijing air quality PM2.5 dataset as an example to demonstrate the analysis process. This dataset measures air pollutants - PM2.5 particles in micrograms per cubic meter over a period of 8 years (2008–2016). It measures the *hourly average of the number of particles that are of size 2.5 microns (PM2.5)* once per hour in Beijing, China.

Let's first import the dataset into R.

```
beijing.pm25<-read.csv("https://umich.instructure.com/files/1823138/download
?download_frd=1")
summary(beijing.pm25)
##      Index            Site       Parameter              Date..LST.
##  Min.   :    1   Beijing:69335   PM2.5:69335   3/13/2011 3:00:    2
##  1st Qu.:17335                                 3/13/2016 3:00:    2
##  Median :34668                                 3/14/2010 3:00:    2
##  Mean   :34668                                 3/8/2009 3:00 :    2
##  3rd Qu.:52002                                 3/8/2015 3:00 :    2
##  Max.   :69335                                 3/9/2014 3:00 :    2
##                                                (Other)       :69323
##       Year          Month             Day              Hour
##  Min.   :2008   Min.   : 1.000   Min.   : 1.00   Min.   : 0.0
##  1st Qu.:2010   1st Qu.: 4.000   1st Qu.: 8.00   1st Qu.: 5.5
##  Median :2012   Median : 6.000   Median :16.00   Median :11.0
##  Mean   :2012   Mean   : 6.407   Mean   :15.73   Mean   :11.5
##  3rd Qu.:2014   3rd Qu.: 9.000   3rd Qu.:23.00   3rd Qu.:17.5
##  Max.   :2016   Max.   :12.000   Max.   :31.00   Max.   :23.0
##
##      Value          Duration        QC.Name
##  Min.   :-999.00   1 Hr:69335   Missing: 4408
##  1st Qu.:  22.00                Valid  :64927
##  Median :  63.00
##  Mean   :  24.99
##  3rd Qu.: 125.00
##  Max.   : 994.00
```

The `Value` column records PM2.5 AQI (Air Quality Index) for 8 years. We observe that there are some missing data in the `Value` column. By looking at the `QC.Name` column, we only have about 6.5% (4408 observations) missing values. One way of solving data-missingness problems, where incomplete observations are recorded, is to replace the absent elements by the corresponding variable mean.

```
beijing.pm25[beijing.pm25$Value==-999, 9]<-NA
beijing.pm25[is.na(beijing.pm25$Value), 9]<-floor(mean(beijing.pm25$Value,
na.rm = T))
```

Here we first reassign the missing values into `NA` labels. Then we replace all `NA` labels with the *mean* computed using all non-missing observations. Note that the `floor()` function casts the arithmetic averages as integer numbers, which is needed as AQI values are expected to be whole numbers.

Now, let's observe the trend of hourly average PM2.5 across 1 day. You can see a significant pattern: The PM2.5 level peeks in the afternoons and is the lowest in the early mornings. It exhibits approximate periodic boundary conditions (these patterns oscillate daily) (Fig. 19.1).

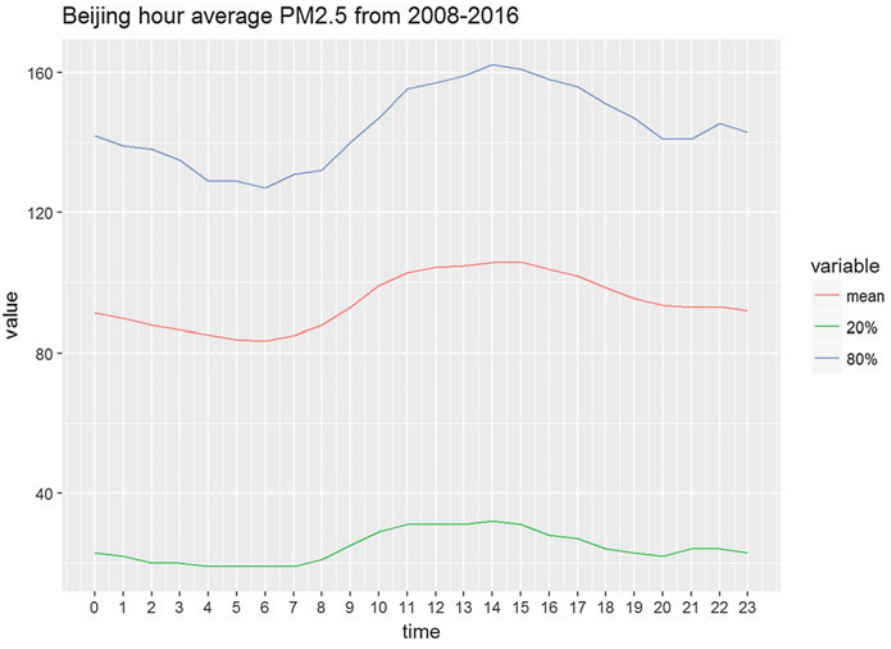

Fig. 19.1 Time course of the mean, top-20%, and bottom-20% air quality in Beijing (PPM2.5)

```
require(ggplot2)

id = 1:nrow(beijing.pm25)
mat = matrix(0,nrow=24,ncol=3)
stat = function(x){
  c(mean(beijing.pm25[iid,"Value"]),quantile(beijing.pm25[iid,"Value"],c(0.2
,0.8)))
}
for (i in 1:24){
  iid = which(id%%24==i-1)
  mat[i,] = stat(iid)
}

mat <- as.data.frame(mat)
colnames(mat) <- c("mean","20%","80%")
mat$time = c(15:23,0:14)
require(reshape2)

## Loading required package: reshape2

dt <- melt(mat,id="time")
colnames(dt)

## [1] "time"     "variable" "value"

ggplot(data = dt,mapping = aes(x=time,y=value,color=variable))+geom_line()+
  scale_x_continuous(breaks = 0:23)+ggtitle("Beijing hour average PM2.5 from
2008-2016")
```

Are there any daily or monthly trends? We can start the data interrogation by building an ARIMA model and examining detailed patterns in the data.

19.1.1 Step 1: Plot Time Series

To begin with, we can visualize the overall trend by plotting PM2.5 values against time. This can be achieved using the `plyr` package.

```
library(plyr)
ts<-ts(beijing.pm25$Value, start=1, end=69335, frequency=1)
ts.plot(ts)
```

The dataset is recorded hourly, and the 8-year time interval includes about 69,335 h of records. Therefore, we start at the first hour and end with 69, 335th h. Each hour has a univariate PM2.5 AQI value measurement, so `frequency=1`.

From this time series plot, Fig. 19.2, we observe that the data has some peaks but most of the AQIs stay under 300 (which is considered hazardous).

The original plot seems have no trend at all. Remember we have our measurements in hours. Will there be any difference if we use monthly average instead of hourly reported values? In this case, we can use Simple Moving Average (SMA) technique to smooth the original graph.

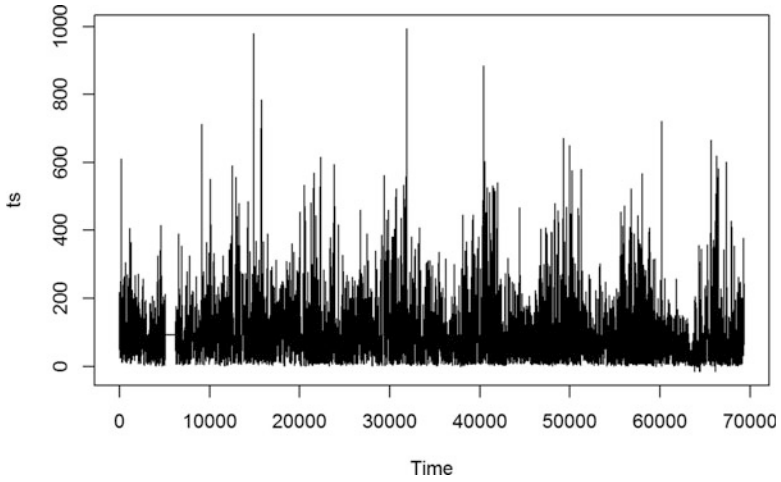

Fig. 19.2 Raw time-series plot of the Beijing air quality measures (2008–2016)

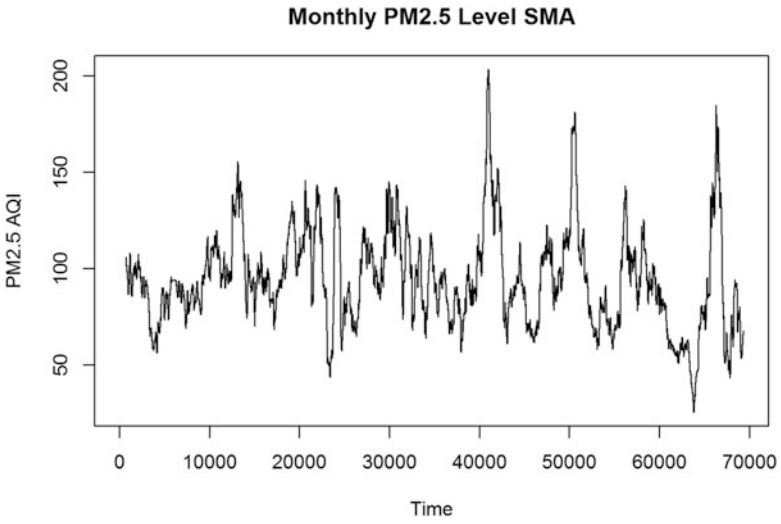

Fig. 19.3 Simple moving monthly average PM2.5 air quality index values

To accomplish this, we need to install the `TTR` package and utilize the SMA() method (Fig. 19.3).

```
#install.packages("TTR")
library(TTR)
bj.month<-SMA(ts, n=720)
plot.ts(bj.month, main="Monthly PM2.5 Level SMA", ylab="PM2.5 AQI")
```

Here we chose `n` to be $24 * 30 = 720$, and we can see some pattern. It seems that for the first 4 years (or approximately 35,040 h), the AQI fluctuates less than the last

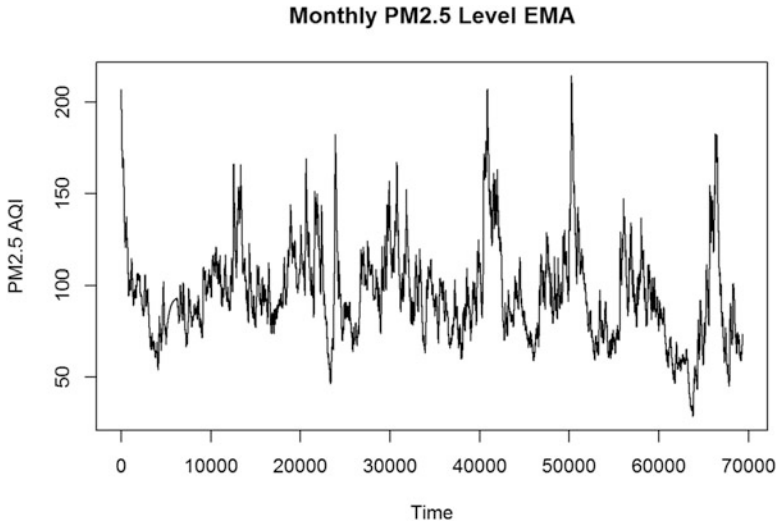

Fig. 19.4 Exponentially-weighted monthly mean of PM2.5 air quality

5 years. Let's see what happens if we use *exponentially-weighted mean*, instead of *arithmetic mean*.

```
bj.month<-EMA(ts, n=1, ratio = 2/(720+1))
plot.ts(bj.month, main="Monthly PM2.5 Level EMA", ylab="PM2.5 AQI")
```

The pattern seems less obvious in this graph, Fig. 19.4. Here we used exponential smoothing ratio of $2/(n + 1)$.

19.1.2 Step 2: Find Proper Parameter Values for ARIMA Model

ARIMA models have 2 components: autoregressive (AR) part and moving average (MA) part. An $ARMA(p, d, q)$ model is a model with p terms in AR, q terms in MA, and d representing the order difference. Differencing is used to make the original dataset approximately stationary. $ARMA(p, d, q)$ has the following analytical form:

$$\left(1 - \sum_{i=1}^{p} \phi_i L^i\right)(1 - L)^d X_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right)\epsilon_t.$$

19.1.3 Check the Differencing Parameter

First, let's try to determine the parameter d. To make the data stationary on the mean (remove any trend), we can use first differencing or second order differencing. Mathematically, first differencing is taking the difference between two adjacent data points:

$$y_t' = y_t - y_{t-1}.$$

While second order differencing is differencing the data twice:

$$y_t^* = y_t' - y_{t-1}' = y_t - 2y_{t-1} + y_{t-2}.$$

Let's see which differencing method is proper for the Beijing PM2.5 dataset. Function `diff()` in R base can be used to calculate differencing. We can plot the differences by `plot.ts()` (Fig. 19.5).

```
par(mfrow= c(2, 1))
bj.diff2<-diff(ts, differences=2)
plot.ts(bj.diff2, main="2nd differencing")
bj.diff<-diff(ts, differences=1)
plot.ts(bj.diff, main="1st differencing")
```

Neither of them appears quite stationary. In this case, we can consider using some smoothing techniques on the data like we just did above (`bj.month<-SMA(ts, n=720)`). Let's see if smoothing by exponentially-weighted mean (EMA) can help making the data approximately stationary (Fig. 19.6).

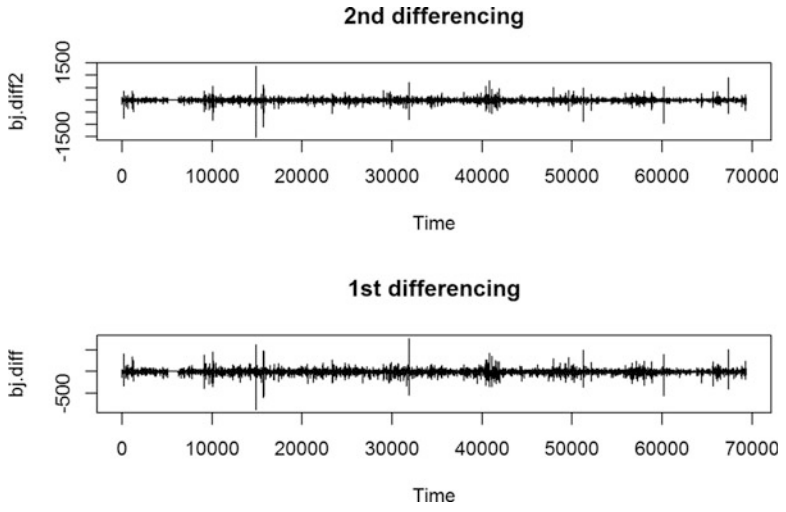

Fig. 19.5 First- and second-order differencing of the AQI data

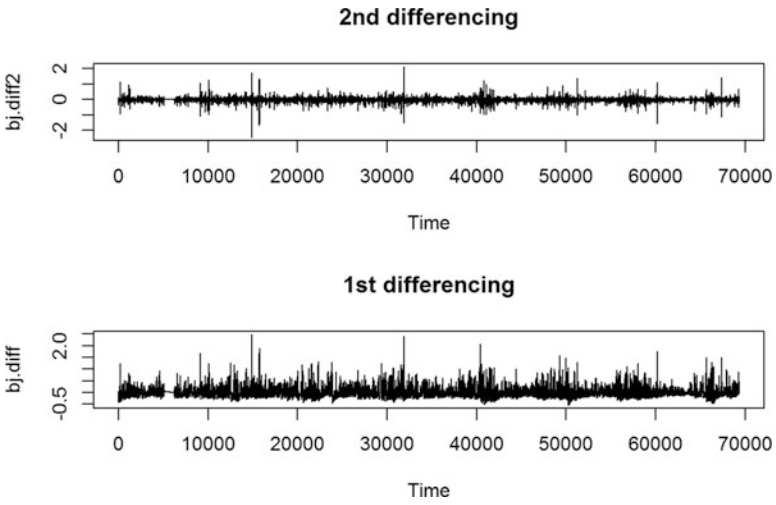

Fig. 19.6 Monthly-smoothed first- and second-order differencing of the AQI data

```
par(mfrow=c(2, 1))
bj.diff2<-diff(bj.month, differences=2)
plot.ts(bj.diff2, main="2nd differencing")
bj.diff<-diff(bj.month, differences=1)
plot.ts(bj.diff, main="1st differencing")
```

Both of these EMA-filtered graphs have tempered variance and appear pretty stationary with respect to the first two moments, mean and variance.

19.1.4 Identifying the AR and MA Parameters

To decide the auto-regressive (AR) and moving average (MA) parameters in the model we need to create **autocorrelation factor (ACF)** and **partial autocorrelation factor (PACF) plots**. PACF may suggest a value for the AR-term parameter q, and ACF may help us determine the MA-term parameter p. We plot the ACF and PACF using the approximately stationary time series, `bj.diff` object (Fig. 19.7).

```
par(mfrow=c(1, 2))
acf(ts(bj.diff), lag.max = 20, main="ACF")
pacf(ts(bj.diff), lag.max = 20, main="PACF")
```

- Pure AR model, (q = 0), will have a cut off at lag p in the PACF.
- Pure MA model, (p = 0), will have a cut off at lag q in the ACF.
- ARIMA(p, q) will (eventually) have a decay in both.

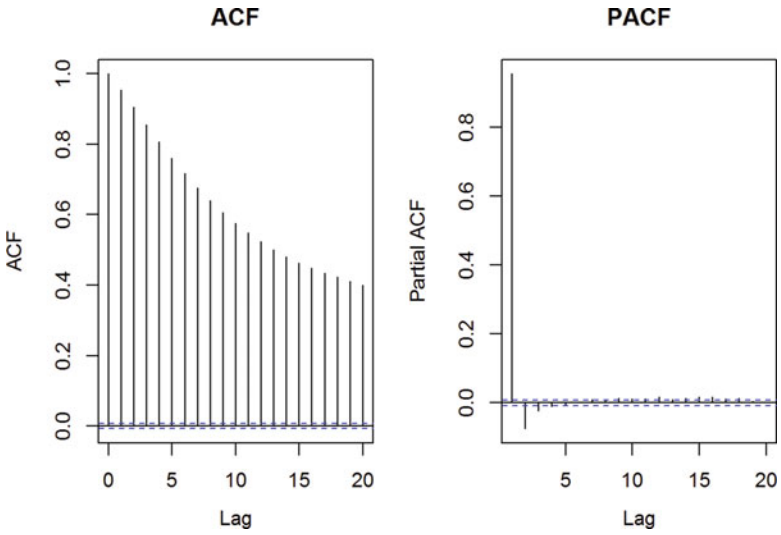

Fig. 19.7 Autocorrelation factor (ACF) and partial autocorrelation factor (PACF) plots of `bj.diff`

All spikes in the plots are outside of the (normal) insignificant zone in the ACF plot while two of them are significant in the PACF plot. In this case, the best ARIMA model is likely to have both AR and MA parts.

We can examine for seasonal effects in the data using `stats::stl()`, a flexible function for decomposing and forecasting the series, which uses averaging to calculate the seasonal component of the series and then subtracts the seasonality. Decomposing the series and removing the seasonality can be done by subtracting the seasonal component from the original series using `forecast::seasadj()`. The frequency parameter in the `ts()` object specifies the periodicity of the data or the number of observations per period, e.g., 30, for monthly smoothed daily data (Fig. 19.8).

```
count_ma = ts(bj.month, frequency=30)
decomp = stl(count_ma, s.window="periodic")
deseasonal_count <- forecast::seasadj(decomp)
plot(decomp)
```

The augmented Dickey-Fuller (ADF) test, `tseries::adf.test` can be used to examine the timeseries stationarity. The *null hypothesis is that the series is non-stationary*. The ADF test quantifies if the change in the series can be explained by a lagged value and a linear trend. Non-stationary series can be *corrected* by differencing to remove trends or cycles.

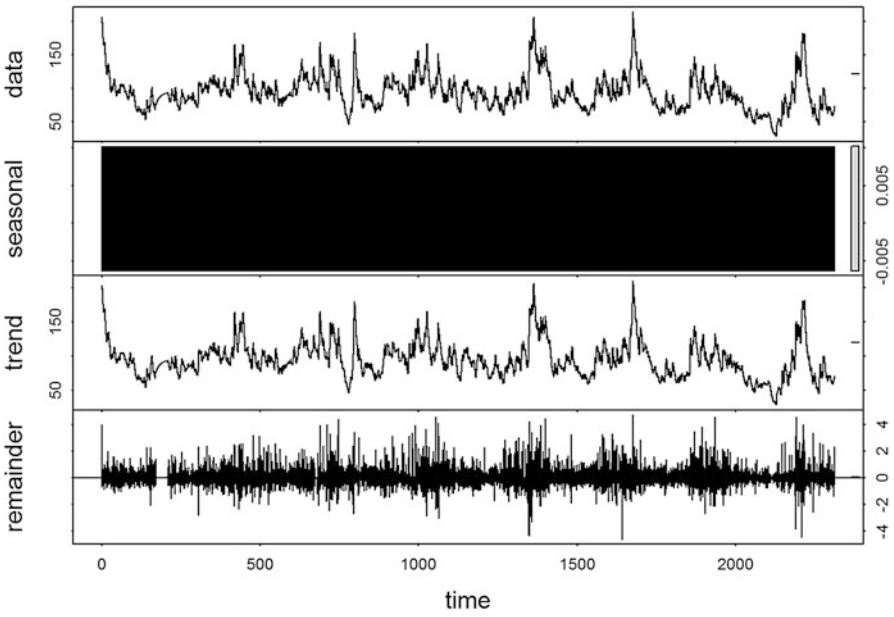

Fig. 19.8 Trend and seasonal decomposition of the time-series

```
tseries::adf.test(count_ma, alternative = "stationary")

##  Augmented Dickey-Fuller Test
##
## data:  count_ma
## Dickey-Fuller = -8.0313, Lag order = 41, p-value = 0.01
## alternative hypothesis: stationary

tseries::adf.test(bj.diff, alternative = "stationary")

##  Augmented Dickey-Fuller Test
##
## data:  bj.diff
## Dickey-Fuller = -29.188, Lag order = 41, p-value = 0.01
## alternative hypothesis: stationary
```

We see that we can reject the null and therefore, there is no statistically significant non-stationarity in the `bj.diff` timeseries.

19.1.5 Step 3: Build an ARIMA Model

As we have some evidence suggesting $d = 1$, the `auto.arima()` function in the `forecast` package can help us to find the optimal estimates for the remaining pair parameters of the ARIMA model, p and q.

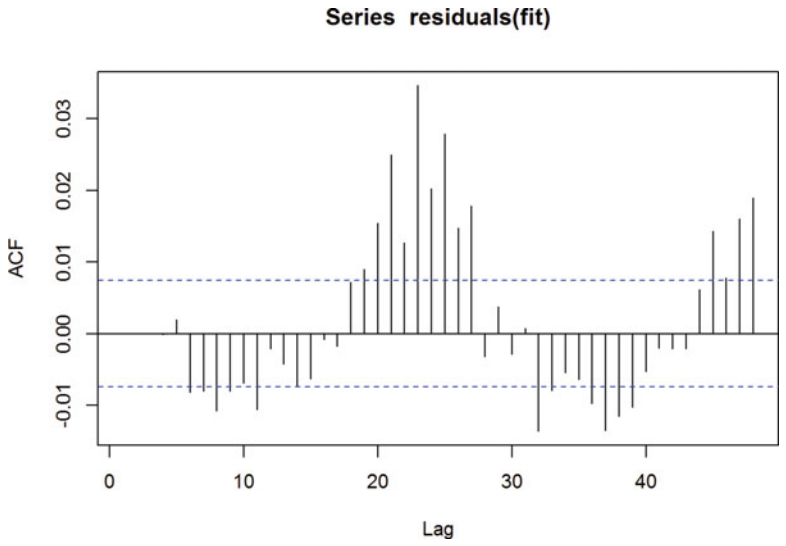

Fig. 19.9 ACF of the time-series residuals

```
# install.packages("forecast")
library(forecast)
fit<-auto.arima(bj.month, approx=F, trace = F)
fit

## Series: bj.month
## ARIMA(1,1,4)
##
## Coefficients:
##          ar1     ma1     ma2     ma3     ma4
##       0.9426  0.0813  0.0323  0.0156  0.0074
## s.e.  0.0016  0.0041  0.0041  0.0041  0.0041
##
## sigma^2 estimated as 0.004604:  log likelihood=88161.91
## AIC=-176311.8   AICc=-176311.8   BIC=-176257

Acf(residuals(fit))
```

Finally, the optimal model determined by the step-wise selection is `ARIMA (1, 1, 4)`. The residual plot is show on Fig. 19.9.

We can also use external information to fit ARIMA models. For example, if we want to add the month information, in case we suspect a seasonal change in PM2.5 AQI, we can use the following script.

```
fit1<-auto.arima(bj.month, xreg=beijing.pm25$Month, approx=F, trace = F)
fit1

## Series: bj.month
## Regression with ARIMA(1,1,4) errors
##
## Coefficients:
##          ar1     ma1     ma2     ma3     ma4  beijing.pm25$Month
##       0.9427  0.0813  0.0322  0.0156  0.0075             -0.0021
## s.e.  0.0016  0.0041  0.0041  0.0041  0.0041              0.0015
##
## sigma^2 estimated as 0.004604:  log likelihood=88162.9
## AIC=-176311.8   AICc=-176311.8   BIC=-176247.8

fit3<-arima(bj.month, order = c(2, 1, 0))
fit3

## Call:
## arima(x = bj.month, order = c(2, 1, 0))
##
## Coefficients:
##          ar1      ar2
##       1.0260  -0.0747
## s.e.  0.0038   0.0038
##
## sigma^2 estimated as 0.004606:  log likelihood = 88138.32,aic=-176270.6
```

We want the model AIC and BIC to be as small as possible. In terms of AIC and BIC, this model is not drastically different compared to the last model without `Month` predictor. Also, the coefficient of `Month` is very small and not significant (according to the t-test) and thus can be removed.

We can examine further the ACF and the PACF plots and the residuals to determine the model quality. When the model order parameters and structure are correctly specified, we expect no significant autocorrelations present in the model residual plots.

```
tsdisplay(residuals(fit), lag.max=45, main='(1,1,4) Model Residuals')
```

There is a clear pattern present in ACF/PACF plots, Fig. 19.10, suggesting that the model residuals repeat with an approximate lag of 12 or 24 months. We may try a modified model with a different parameters, e.g., $p = 24$ or $q = 24$. We can define a new `displayForecastErrors()` function to show a histogram of the forecasted errors (Figs. 19.11 and 19.12).

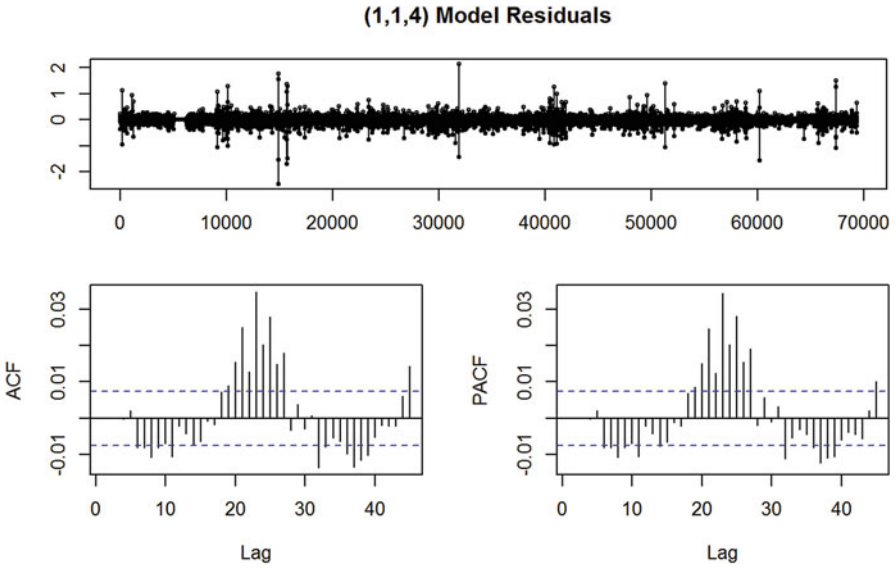

Fig. 19.10 ARIMA(1,1,4) model plot, ACF and PACF plots of the resiguals for `bj.month`

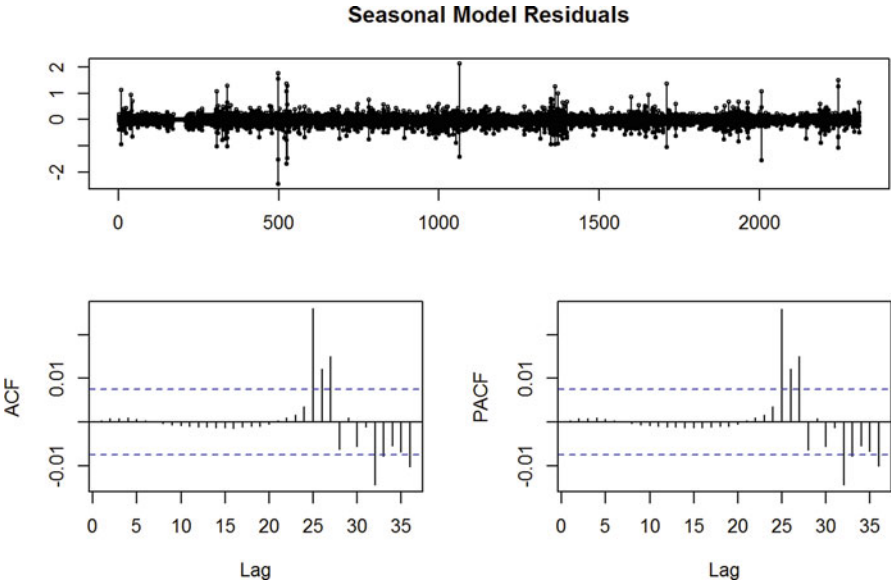

Fig. 19.11 An improved ARIMA(1,1,24) model plot, ACF and PACF plots of the resiguals for `bj.month`

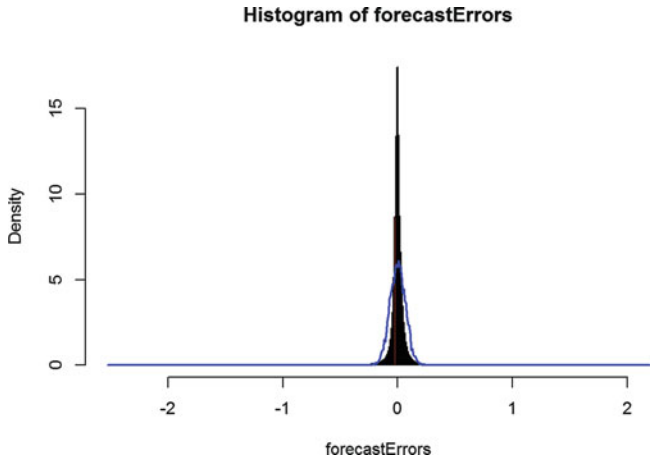

Fig. 19.12 Diagnostic plot of the residuals of the ARIMA(1,1,24) time-series model for `bj.month`

```
fit24 <- arima(deseasonal_count, order=c(1,1,24)); fit24

## Call:
## arima(x = deseasonal_count, order = c(1, 1, 24))
##
## Coefficients:
##          ar1     ma1     ma2     ma3      ma4      ma5      ma6      ma7
##       0.9496  0.0711  0.0214  0.0054  -0.0025  -0.0070  -0.0161  -0.0149
## s.e.  0.0032  0.0049  0.0049  0.0048   0.0047   0.0046   0.0045   0.0044
##           ma8      ma9     ma10     ma11     ma12     ma13     ma14
##       -0.0162  -0.0118  -0.0100  -0.0136  -0.0045  -0.0055  -0.0075
## s.e.   0.0044   0.0043   0.0042   0.0042   0.0042   0.0041   0.0041
##          ma15     ma16     ma17     ma18     ma19     ma20     ma21     ma22
##       -0.0060  -0.0005  -0.0019  0.0066  0.0088  0.0156  0.0247  0.0117
## s.e.   0.0041   0.0041   0.0041  0.0041  0.0041  0.0040  0.0040  0.0040
##          ma23     ma24
##       0.0319  0.0156
## s.e.  0.0040  0.0039
##
## sigma^2 estimated as 0.004585:log likelihood = 88295.88,aic = -176539.8

tsdisplay(residuals(fit24), lag.max=36, main='Seasonal Model Residuals')
```

```
displayForecastErrors <- function(forecastErrors)
  {
     # Generate a histogram of the Forecast Errors
     binsize <- IQR(forecastErrors)/4
     sd   <- sd(forecastErrors)
     min  <- min(forecastErrors) - sd
     max  <- max(forecastErrors) + sd

     # Generate 5K normal(0,sd) RVs
     norm <- rnorm(5000, mean=0, sd=sd)
     min2 <- min(norm)
     max2 <- max(norm)
```

```
    if (min2 < min) { min <- min2 }
    if (max2 > max) { max <- max2 }

    # Plot red histogram of the forecast errors
    bins <- seq(min, max, binsize)
    hist(forecastErrors, col="red", freq=FALSE, breaks=bins)

    myHist <- hist(norm, plot=FALSE, breaks=bins)

    # Overlay the Blue normal curve on top of forecastErrors histogram
    points(myHist$mids, myHist$density, type="l", col="blue", lwd=2)
}

displayForecastErrors(residuals(fit24))
```

19.1.6 Step 4: Forecasting with ARIMA Model

Now, we can use our models to make predictions for future PM2.5 AQI. We will use the function `forecast()` to make predictions. In this function, we have to specify the number of periods we want to forecast. Using the smoothed data, we can make predictions for the next month, July 2016. As each month has about $24 \times 30 = 720$ h, we specify a horizon `h = 720` (Fig. 19.13).

```
par(mfrow=c(1, 1))
ts.forecasts<-forecast(fit, h=720)
plot(ts.forecasts, include = 2880)
```

When plotting the forecasted values with the original smoothed data, we include only the last 3 months in the original smoothed data to see the predicted values clearer. The shaded regions indicate ranges of expected errors. The darker (inner) region represents by *80% confidence range* and the lighter (outer) region bounds by the

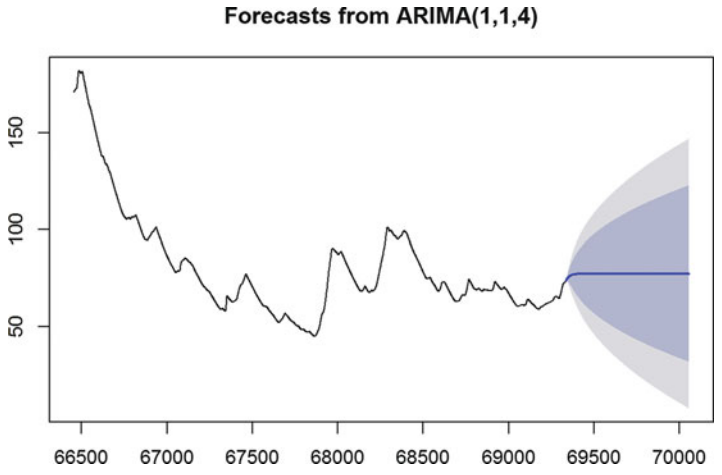

Fig. 19.13 Prospective out-of-range prediction intervals of the ARIMA(1,1,4) time-series model

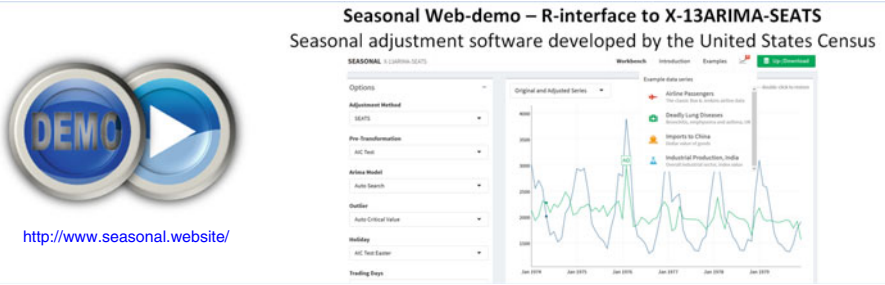

Fig. 19.14 Live Demo: Interactive US Census ARIMA modeling

95% interval. Obviously near-term forecasts have tighter ranges of expected errors, compared to longer-term forecasts where the variability naturally expands. A live demo of US Census data is shown on Fig. 19.14.

19.2 Structural Equation Modeling (SEM)-Latent Variables

Timeseries analyses provide effective strategies to interrogate longitudinal univariate data. What happens if we have multiple, potentially associated, measurements recorded at each time point?

SEM is a general multivariate statistical analysis technique that can be used for causal modeling/inference, path analysis, confirmatory factor analysis (CFA), covariance structure modeling, and correlation structure modeling. This method allows *separation of observed and latent variables*. Other standard statistical procedures may be viewed as special cases of SEM, where statistical significance may be less important, and covariances are the core of structural equation models.

Latent variables are features that are not directly observed but may be inferred from the actually observed variables. In other words, a combination or transformation of observed variables can create latent features, which may help us reduce the dimensionality of data. Also, SEM can address multi-collinearity issues when we fit models because we can combine some high collinearity variables to create a single (latent) variable, which can then be included into the model.

19.2.1 Foundations of SEM

SEMs consist of two complementary components: (1) a *path model*, quantifying specific cause-and-effect relationships between observed variables, and (2) a *measurement model*, quantifying latent linkages between unobservable components and observed variables. The LISREL (LInear Structural RELations) framework represents a unifying mathematical strategy to specify these linkages, see Grace 2006.

The most general kind of SEM is a structural regression path model with latent variables, which account for measurement errors of observed variables. *Model identification* determines whether the model allows for unique parameter estimates and may be based on model degrees of freedom ($df_M \geq 0$) or a known scale for every latent feature. If ν represents the number of observed variables, then the total degrees of freedom for a SEM, $\frac{\nu(1+\nu)}{2}$, corresponds to the number of variances and unique covariances in a variance-covariance matrix for all the features, and the model degrees of freedom, $df_M = \frac{\nu(1+\nu)}{2} - l$, where l is the number of estimated parameters.

Examples include:

- *Just-identified model* ($df_M = 0$) with unique parameter estimates,
- *Over-identified model* ($df_M > 0$) desirable for model testing and assessment,
- *Under-identified model* ($df_M < 0$) is not guaranteed unique solutions for all parameters. In practice, such models occur when the effective degrees of freedom are reduced due to two or more highly-correlated features, which presents problems with parameter estimation. In these situations, we can exclude or combine some of the features boosting the degrees of freedom.

The latent variables' *scale* property reflects their unobservable, not measurable, characteristics. The latent scale, or unit, may be inferred from one of its observed constituent variables, e.g., by imposing a unit loading identification constraint fixing at 1.0 the factor loading of one observed variable.

An SEM model with appropriate *scale* and degrees of freedom conditions may be identifiable subject to Bollen's two-step identification rule. When both the CFA path components of the SEM model are identifiable, then the whole SR model is identified, and model fitting can be initiated.

- For the confirmatory factor analysis (CFA) part of the SEM, identification requires (1) a minimum of two observed variables for each latent feature, (2) independence between measurement errors and the latent variables, and (3) independence between measurement errors.
- For the path component of the SEM, ignoring any observed variables used to measure latent variables, model identification requires: (1) errors associated with endogenous latent variables to be uncorrelated, and (2) all causal effects to be unidirectional.

The LISREL representation can be summarized by the following matrix equations:

$$\text{measurement model component} \begin{cases} x = \Lambda_x \xi + \delta, \\ y = \Lambda_y \eta + \epsilon. \end{cases}$$

And

$$\text{path model component } \eta = B\eta + \Gamma\xi + \zeta,$$

where:

- $x_{p \times 1}$ is a vector of observed *exogenous variables* representing a linear function of $\xi_{j \times 1}$, vector of *exogenous latent variables*,
- $\delta_{p \times 1}$ is a *vector of measurement error*, Λ_x is a $p \times j$ matrix of factor loadings relating x to ξ,
- $y_{q \times 1}$ is a vector of observed *endogenous variables*,
- $\eta_{k \times 1}$ is a vector of *endogenous latent variables*,
- $\epsilon_{q \times 1}$ is a vector of *measurement error for the endogenous variables*, and
- Λ_y is a $q \times k$ matrix of factor loadings relating y to η.

Let's also denote the two variance-covariance matrices, $\Theta_\delta(p \times p)$ and $\Theta_\epsilon(q \times q)$, representing the variance-covariance matrices among the measurement errors δ and ϵ, respectively. The third equation describing the LISREL path model component as relationships among latent variables includes:

- $B_{k \times k}$ a matrix of path coefficients describing the *relationships among endogenous latent variables*,
- $\Gamma_{k \times j}$ as a matrix of path coefficients representing the *linear effects of exogenous variables on endogenous variables*,
- $\zeta_{k \times 1}$ as a vector of *errors of endogenous variables*, and the corresponding two variance-covariance matrices $\Phi_{j \times j}$ of the *latent exogenous variables*, and
- $\Psi_{k \times k}$ of the *errors of endogenous variables*.

The basic statistic for a typical SEM implementation is based on covariance structure modeling and model fitting relies on optimizing an objective function, $\min\{f(\Sigma, S)\}$, representing the difference between the model-implied variance-covariance matrix, Σ, predicted from the causal and non-causal associations specified in the model, and the corresponding observed variance-covariance matrix S, which is estimated from observed data. The objective function, $f(\Sigma, S)$ can be estimated as shown below, see Shipley 2016.

In general, causation implies correlation, suggesting that if there is a causal relationship between two variables, there must also be a systematic relationship between them. Specifying a set of theoretical causal paths, we can reconstruct the model-implied variance-covariance matrix, Σ, from total effects and unanalyzed associations. The LISREL strategy specifies the following mathematical representation:

$$\Sigma = \begin{vmatrix} \Lambda_y A(\Gamma\Phi\Gamma' + \Psi)A'\Lambda'_y + \Theta_\epsilon & \Lambda_y A\Gamma\Phi\Lambda'_x \\ \Lambda_x\Phi\Gamma'A'\Lambda'_y & \Lambda_x\Phi\Lambda'_x + \Theta_\delta \end{vmatrix},$$

where $A = (I - B)^{-1}$. This representation of Σ does not involve the observed and latent exogenous and endogenous variables, x, y, ξ, η. Maximum likelihood estimation (MLE) may be used to obtain the Σ parameters via iterative searches for a set of optimal parameters minimizing the element-wise deviations between Σ and S.

The process of optimizing the objective function $f(\Sigma, S)$ can be achieved by computing the log likelihood ratio, i.e., comparing the likelihood of a given fitted model to the likelihood of a perfectly fit model. MLE estimation requires

multivariate normal distribution for the endogenous variables and Wishart distribution for the observed variance-covariance matrix, S.

Using MLE estimation simplifies the objective function to:

$$f(\Sigma, S) = \ln \mid \Sigma \mid + tr\left(S \times \Sigma^{-1}\right) - \ln \mid S \mid - tr\left(SS^{-1}\right),$$

where $tr()$ is the trace of a matrix. The optimization of $f(\Sigma, S)$ also requires independent and identically distributed observations and positive definite matrices, Σ, S. The iterative MLE optimization generates estimated variance-covariance matrices and path coefficients for the specified model. More details on model assessment (using Root Mean Square Error of Approximation, RMSEA, and Goodness of Fit Index) and the process of defining a priori SEM hypotheses are available in Lam & Maguire, 2012.

19.2.2 SEM Components

The R `Lavaan` package uses the following SEM syntax, Table 19.1, to represent relationships between variables. We can follow the following table to specify `Lavaan` models:

For example in R we can write the following model `model<-`

`' # regressions`

$$y1 + y2 \sim f1 + f2 + x1 + x2$$
$$f1 \sim f2 + f3$$
$$f2 \sim f3 + x1 + x2$$

`# latent variable definitions`

$$f1 =\sim y1 + y2 + y3$$
$$f2 =\sim y4 + y5 + y6$$
$$f3 =\sim y7 + y8 + y9 + y10$$

`# variances and covariances`

$$y1 \sim\sim y1$$
$$y1 \sim\sim y2$$
$$f1 \sim\sim f2$$

`# intercepts`

$$y1 \sim 1$$
$$f1 \sim 1$$

`'`

Note that the two `"'"` symbols (in the beginning and ending of a model description) are very important in the `R`-syntax.

Table 19.1 Lavaan syntax for specifying the relations between variables and their variance-covariance structure

Formula type	Operator	Explanation
Latent variable definition	=~	Is measured by
Regression	~	Is regressed on
(Residual) (co)variance	~~	Is correlated with
Intercept	~1	Intercept

19.2.3 Case Study – Parkinson's Disease (PD)

Let's use the PPMI dataset in our class file as an example to illustrate SEM model fitting.

Step 1 – Collecting Data

The Parkinson's Disease Data represents a realistic simulation case-study to examine associations between clinical, demographic, imaging and genetics variables for Parkinson's disease. This is an example of Big Data for investigating important neurodegenerative disorders.

Step 2 – Exploring and Preparing the Data

Now, we can import the dataset into R and recode the `ResearchGroup` variable into a binary variable.

```
par(mfrow=c(1, 1))
PPMI<-read.csv("https://umich.instructure.com/files/330397/download?download
_frd=1")
summary(PPMI)

##     FID_IID     L_insular_cortex_ComputeArea L_insular_cortex_Volume
##  Min.   :3001   Min.   :  50.03              Min.   :   22.63
##  1st Qu.:3272   1st Qu.:1976.88              1st Qu.: 4881.36
##  Median :3476   Median :2498.65              Median : 7236.76
##  Mean   :3534   Mean   :2255.20              Mean   : 6490.84
##  3rd Qu.:3817   3rd Qu.:2744.05              3rd Qu.: 8405.43
##  Max.   :4139   Max.   :3650.81              Max.   :13499.92
…
##   UPDRS_part_I    UPDRS_part_II    UPDRS_part_III    time_visit
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.00   Min.   : 0.00
##  1st Qu.: 0.000   1st Qu.: 2.000   1st Qu.:12.00   1st Qu.: 8.25
##  Median : 1.000   Median : 5.000   Median :20.00   Median :21.00
##  Mean   : 1.286   Mean   : 6.087   Mean   :19.44   Mean   :23.50
##  3rd Qu.: 2.000   3rd Qu.: 9.000   3rd Qu.:27.00   3rd Qu.:37.50
##  Max.   :13.000   Max.   :28.000   Max.   :61.00   Max.   :54.00
##  NA's   :549      NA's   :553      NA's   :554

PPMI$ResearchGroup<-ifelse(PPMI$ResearchGroup=="Control", "1", "0")
```

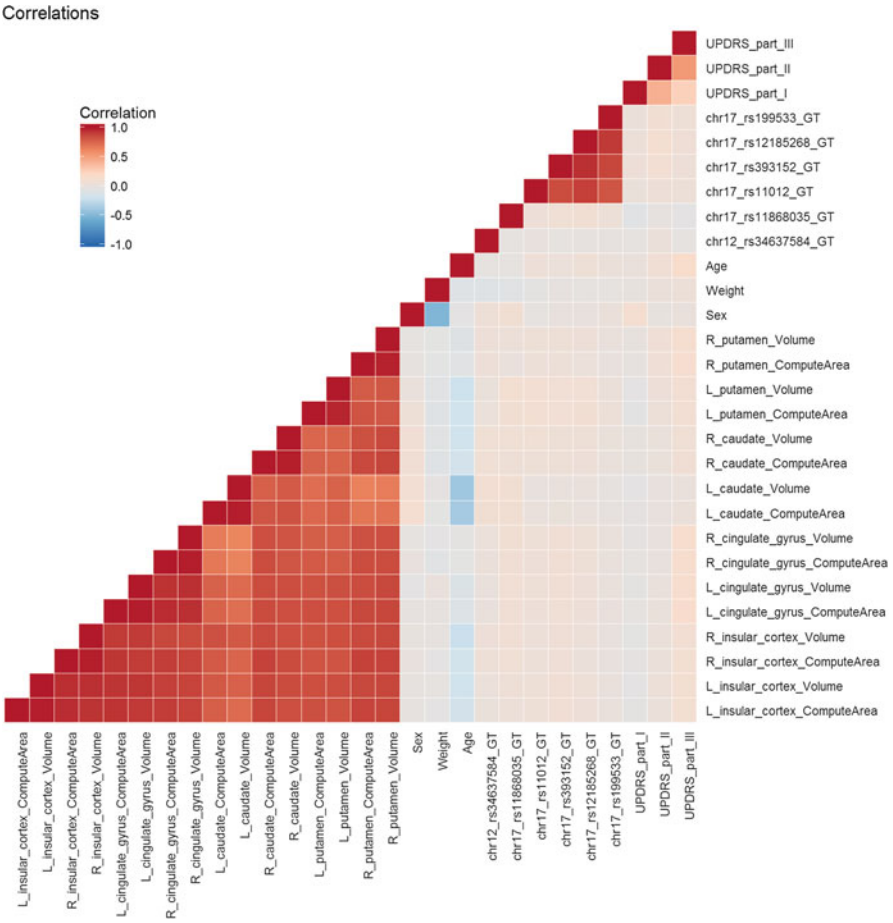

Fig. 19.15 Pair-wise correlation structure of the Parkinson's disease (PPMI) data.

This large dataset has 1,746 observations and 31 variables with missing data in some of them. A lot of the variables are highly correlated. You can inspect high correlation using *heat maps*, which reorders these covariates according to correlations to illustrate clusters of high-correlations (Fig. 19.15).

```
pp_heat <- PPMI[complete.cases(PPMI),-20]
corr_mat = cor(pp_heat)
# Remove upper triangle
corr_mat_lower = corr_mat
corr_mat_lower[upper.tri(corr_mat_lower)] = NA
# Melt correlation matrix and make sure order of factor variables is correct
corr_mat_melted = melt(corr_mat_lower)
colnames(corr_mat_melted) <- c("Var1", "Var2", "value")
corr_mat_melted$Var1 = factor(corr_mat_melted$Var1, levels=colnames(corr_mat
))
corr_mat_melted$Var2 = factor(corr_mat_melted$Var2, levels=colnames(corr_mat
))
```

```
# Plot
corr_plot = ggplot(corr_mat_melted, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile(color='white') +
  scale_fill_distiller(limits=c(-1, 1), palette='RdBu', na.value='white',
                       name='Correlation') +
  ggtitle('Correlations') +
  coord_fixed(ratio=1) +
  theme_minimal() +
  scale_y_discrete(position="right") +
  theme(axis.text.x=element_text(angle=45, vjust=1, hjust=1),
        axis.title.x=element_blank(),
        axis.title.y=element_blank(),
        panel.grid.major=element_blank(),
        legend.position=c(0.1,0.9),
        legend.justification=c(0,1))
corr_plot
```

And here are some specific correlations

```
cor(PPMI$L_insular_cortex_ComputeArea, PPMI$L_insular_cortex_Volume)

## [1] 0.9837297

cor(PPMI$UPDRS_part_I, PPMI$UPDRS_part_II, use = "complete.obs")

## [1] 0.4027434

cor(PPMI$UPDRS_part_II, PPMI$UPDRS_part_III, use = "complete.obs")

## [1] 0.5326681
```

One way to solve this substantial multivariate correlation issue is to create some latent variables. We can consider the following model.

```
model1<-
  '
    Imaging =~ L_cingulate_gyrus_ComputeArea  + L_cingulate_gyrus_Volume+R_c
ingulate_gyrus_ComputeArea+R_cingulate_gyrus_Volume+R_insular_cortex_Compute
Area+R_insular_cortex_Volume
    UPDRS=~UPDRS_part_I+UPDRS_part_II+UPDRS_part_III
    DemoGeno =~ Weight+Sex+Age

    ResearchGroup ~ Imaging + DemoGeno + UPDRS
  '
```

Here we try to create three latent variables: `Imaging`, `DemoGeno`, and `UPDRS`. Let's fit a SEM model using `cfa()`, a confirmatory factor analysis function. Before fitting the data, we need to scale them. However, we don't need to scale our binary response variable. We can use the following code for normalizing the data.

```
mydata<-scale(PPMI[, -20])
mydata<-data.frame(mydata, PPMI$ResearchGroup)
colnames(mydata)[31]<-"ResearchGroup"
```

Step 3 – Fitting a Model on the Data

Now, we can start to build the model. The `cfa()` function we will use is part of the `lavaan` package.

```
# install.packages("lavaan")
library(lavaan)

fit<-cfa(model1, data=mydata, missing = 'FIML')
```

Here we can see some warning messages. Both our covariance and error term matrices are not positive definite. Non-positive definite matrices can cause the estimates of our model to be biased. There are many factors that can lead to this problem. In this case, we might create some latent variables that are not a good fit for our data. Let's try to delete the `DemoGeno` latent variable. We can add `Weight`, `Sex`, and `Age` directly to the regression model.

```
model2 <-
    '
# (1) Measurement Model
Imaging =~  L_cingulate_gyrus_ComputeArea  + L_cingulate_gyrus_Volume+R_cing
ulate_gyrus_ComputeArea+R_cingulate_gyrus_Volume+R_insular_cortex_ComputeAre
a+R_insular_cortex_Volume
UPDRS =~  UPDRS_part_I  +UPDRS_part_II + UPDRS_part_III
# (2) Regressions
ResearchGroup ~ Imaging + UPDRS +Age+Sex+Weight
'
```

When fitting `model2`, the warning messages are gone. We can see that falsely adding a latent variable can cause those matrices to be not positive definite. Currently, the `lavaan` functions `sem()` and `cfa()` are the same.

```
fit<-cfa(model2, data=mydata, missing = 'FIML')
summary(fit, fit.measures=TRUE)
## lavaan (0.5-23.1097) converged normally after 107 iterations
##
##   Number of observations                          1764
##
##   Number of missing patterns                         4
##
##   Estimator                                         ML
##   Minimum Function Test Statistic             7714.119
##   Degrees of freedom                                60
##   P-value (Chi-square)                           0.000
##
## Model test baseline model:
##
##   Minimum Function Test Statistic           30237.866
##   Degrees of freedom                               75
##   P-value                                       0.000
```

```
##
## User model versus baseline model:
##
##   Comparative Fit Index (CFI)                    0.746
##   Tucker-Lewis Index (TLI)                       0.683
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)                     NA
##   Loglikelihood unrestricted model (H1)             NA
##
##   Number of free parameters                         35
##   Akaike (AIC)                                      NA
##   Bayesian (BIC)                                    NA
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.269
##   90 Percent Confidence Interval          0.264  0.274
##   P-value RMSEA <= 0.05                          0.000
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.052
##
## Parameter Estimates:
##
##   Information                                 Observed
##   Standard Errors                             Standard
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   Imaging =~
##     L_cnglt_gyr_CA    1.000
##     L_cnglt_gyrs_V    0.994    0.004  260.366    0.000
##     R_cnglt_gyr_CA    0.961    0.007  134.531    0.000
##     R_cnglt_gyrs_V    0.955    0.008  126.207    0.000
##     R_nslr_crtx_CA    0.930    0.009  101.427    0.000
##     R_nslr_crtx_Vl    0.920    0.010   94.505    0.000
##   UPDRS =~
##     UPDRS_part_I      1.000
##     UPDRS_part_II     1.890    0.177   10.699    0.000
##     UPDRS_part_III    2.345    0.248    9.468    0.000
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   ResearchGroup ~
##     Imaging           0.008    0.010    0.788    0.431
##     UPDRS            -0.828    0.080  -10.299    0.000
##     Age               0.019    0.009    2.121    0.034
##     Sex              -0.010    0.010   -0.974    0.330
##     Weight            0.005    0.010    0.481    0.631
```

```
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   Imaging ~~
##     UPDRS             0.059    0.014    4.361    0.000
##
## Intercepts:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .L_cnglt_gyr_CA   -0.000    0.024   -0.001    1.000
##    .L_cnglt_gyrs_V   -0.000    0.024   -0.001    1.000
##    .R_cnglt_gyr_CA   -0.000    0.024   -0.001    1.000
##    .R_cnglt_gyrs_V   -0.000    0.024   -0.001    1.000
##    .R_nslr_crtx_CA   -0.000    0.024   -0.001    1.000
##    .R_nslr_crtx_Vl   -0.000    0.024   -0.001    1.000
##    .UPDRS_part_I     -0.135    0.032   -4.225    0.000
##    .UPDRS_part_II    -0.255    0.033   -7.621    0.000
##    .UPDRS_part_III   -0.317    0.034   -9.181    0.000
##    .ResearchGroup     1.290    0.011  119.239    0.000
##     Imaging           0.000
##     UPDRS             0.000
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .L_cnglt_gyr_CA    0.006    0.001    9.641    0.000
##    .L_cnglt_gyrs_V    0.019    0.001   23.038    0.000
##    .R_cnglt_gyr_CA    0.083    0.003   27.917    0.000
##    .R_cnglt_gyrs_V    0.093    0.003   27.508    0.000
##    .R_nslr_crtx_CA    0.141    0.005   28.750    0.000
##    .R_nslr_crtx_Vl    0.159    0.006   28.728    0.000
##    .UPDRS_part_I      0.877    0.038   23.186    0.000
##    .UPDRS_part_II     0.561    0.033   16.873    0.000
##    .UPDRS_part_III    0.325    0.036    9.146    0.000
##    .ResearchGroup     0.083    0.006   14.808    0.000
##     Imaging           0.993    0.034   29.509    0.000
##     UPDRS             0.182    0.035    5.213    0.000
```

19.2.4 Outputs of Lavaan SEM

In the output of our model, we have information about how to create these two latent variables (`Imaging`, `UPDRS`) and the estimated regression model. Specifically, it gives the following information.

1. First six lines are called the header contains the following information:
   - Lavaan version number.
   - Lavaan convergence information (normal or not), and #number of iterations needed.
   - The number of observations that were effectively used in the analysis.
   - The estimator that was used to obtain the parameter values (here: ML).
   - The model test statistic, the degrees of freedom, and a corresponding p-value.
2. Next, we have the Model test baseline model and the value for the SRMR

3. The last section contains the parameter estimates, standard errors (if the information matrix is expected or observed, and if the standard errors are standard, robust, or based on the bootstrap). Then, it tabulates all free (and fixed) parameters that were included in the model. Typically, first the latent variables are shown, followed by covariances and (residual) variances. The first column (Estimate) contains the (estimated or fixed) parameter value for each model parameter; the second column (Std.err) contains the standard error for each estimated parameter; the third column (Z-value) contains the Wald statistic (which is simply obtained by dividing the parameter value by its standard error); and the last column contains the p-value for testing the null hypothesis that the parameter equals zero in the population.

19.3 Longitudinal Data Analysis-Linear Mixed Models

As mentioned earlier, longitudinal studies take measurements for the same individual repeatedly through a period of time. Under this setting, we can measure the change after a specific treatment. However, the measurements for the same individual may be correlated with each other. Thus, we need special models that deal with this type of internal multivariate dependencies.

If we use the latent variable UPDRS (created in the output of SEM model) rather than the research group as our response we can obtain a longitudinal analysis model. In longitudinal analysis, time is often an important model variable.

19.3.1 Mean Trend

According to the output of model `fit`, our latent variable `UPDRS` is a combination of three observed variables-`UPDRS_part_I`, `UPDRS_part_II`, and `UPDRS_part_III`. We can visualize how average `UPDRS` values differ among the research groups over time.

```
mydata$UPDRS<-mydata$UPDRS_part_I+1.890*mydata$UPDRS_part_II+2.345*mydata$UP
DRS_part_III
mydata$Imaging<-mydata$L_cingulate_gyrus_ComputeArea  +0.994*mydata$L_cingul
ate_gyrus_Volume+0.961*mydata$R_cingulate_gyrus_ComputeArea+0.955*mydata$R_c
ingulate_gyrus_Volume+0.930*mydata$R_insular_cortex_ComputeArea+0.920*mydata
$R_insular_cortex_Volume
```

The above code stores the latent `UPDRS` and `Imaging` variables into `mydata`. By now, we are experienced with using the package `ggplot2` for data visualization. Now, we will use it to set the x and y axes as `time` and `UPDRS`, and then display the trend of the individual level `UPDRS`.

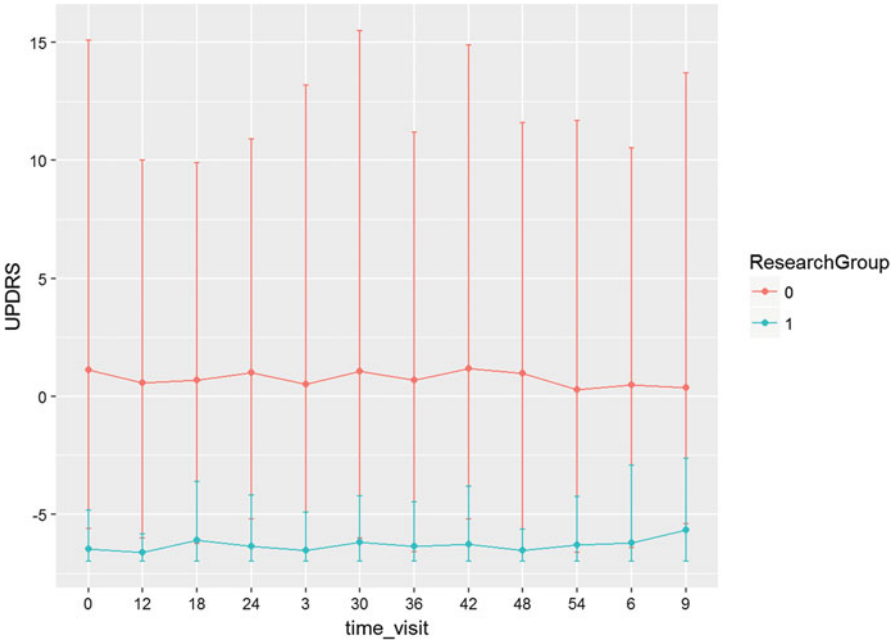

Fig. 19.16 Average UPDRS scores of the two cohorts in the PPMI dataset, patients (1) and controls (0)

```
require(ggplot2)
p<-ggplot(data=mydata, aes(x=time_visit, y=UPDRS, group=FID_IID))
dev.off()

p+geom_point()+geom_line()
```

This graph is a bit messy without a clear pattern emerging. Let's see if group-level graphs may provide more intuition. We will use the `aggregate()` function to get the mean, minimum and maximum of `UPDRS` for each time point. Then, we will use separate color for the two research groups and examine their mean trends (Fig. 19.16).

```
ppmi.mean<-aggregate(UPDRS~time_visit+ResearchGroup, FUN = mean, data=
mydata[, c(30, 31, 32)])
ppmi.min<-aggregate(UPDRS~time_visit+ResearchGroup, FUN = min, data=
mydata[, c(30, 31, 32)])
ppmi.max<-aggregate(UPDRS~time_visit+ResearchGroup, FUN = max, data=
mydata[, c(30, 31, 32)])
ppmi.boundary<-merge(ppmi.min, ppmi.max,by=c("time_visit","ResearchGroup"))
ppmi.all<-merge(ppmi.mean,ppmi.boundary,by=c("time_visit","ResearchGroup"))
pd <- position_dodge(0.1)
p1<-ggplot(data=ppmi.all, aes(x=time_visit, y=UPDRS, group=ResearchGroup,
colour=ResearchGroup))
p1+geom_errorbar(aes(ymin=UPDRS.x, ymax=UPDRS.y, width=0.1))+geom_point()+
geom_line()
```

Despite slight overlaps in some lines, the resulting graph illustrates better the mean differences between the two cohorts. The control group (1) appears to have relative lower means and tighter ranges compared to the PD patient group (0). However, we need further data interrogation to determine if this visual (EDA) evidence translates into statistically significant group differences.

Generally speaking we can always use the *General Linear Modeling (GLM)* framework. However, GLM may ignore the individual differences. So, we can try to fit a *Linear Mixed Model (LMM)* to incorporate different intercepts for each individual participant. Consider the following GLM:

$$\begin{aligned} UPDRS_{ij} \sim \beta_0 + \beta_1 * Imaging_{ij} + \beta_2 * ResearchGroup_i + \beta_3 * timeVisit_j \\ + \beta_4 * ResearchGroup_i * time_v isit_j + \beta_5 * Age_i + \beta_6 * Sex_i \\ + \beta_7 * Weight_i + \epsilon_{ij}. \end{aligned}$$

If we fit a different intercept, b_i, for each individual (indicated by FID_IID), we obtain the following LMM model:

$$\begin{aligned} UPDRS_{ij} \sim \beta_0 + \beta_1 * Imaging + \beta_2 * ResearchGroup + \beta_3 * timeVisit_j \\ + \beta_4 * ResearchGroup_i * timeVisit_j + \beta_5 * Age_i + \beta_6 * Sex_i \\ + \beta_7 * Weight_i + b_i + \epsilon_{ij}. \end{aligned}$$

The LMM actually has two levels:

Stage 1

$$Y_i = Z_i \beta_i + \epsilon_i,$$

where both Z_i and β_i are matrices.

Stage 2
The second level allows fitting random effects in the model.

$$\beta_i = A_i * \beta + b_i.$$

So, the full model in matrix form would be:

$$Y_i = X_i * \beta + Z_i * b_i + \epsilon_i.$$

In this case study, we only consider random intercept and avoid including random slopes, however the model can indeed be extended. In other words, $Z_i = 1$ in our simple model. Let's compare the two models (GLM and LMM). One R package implementing LMM is `lme4`.

```
#install.packages("lme4")
#install.packages("arm")
library(lme4)

library(arm)

#GLM
model.glm<-glm(UPDRS~Imaging+ResearchGroup*time_visit+Age+Sex+Weight,data=my
data)
summary(model.glm)
```

```
##
## Call:
## glm(formula = UPDRS ~ Imaging + ResearchGroup * time_visit +
##     Age + Sex + Weight, data = mydata)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -7.6065  -2.4581  -0.3159   1.8328  14.9746
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)                0.70000    0.10844   6.455 1.57e-10 ***
## Imaging                    0.03834    0.01893   2.025   0.0431 *
## ResearchGroup1            -6.93501    0.33445 -20.736  < 2e-16 ***
## time_visit                 0.05077    0.10843   0.468   0.6397
## Age                        0.54171    0.10839   4.998 6.66e-07 ***
## Sex                        0.16170    0.11967   1.351   0.1769
## Weight                     0.20980    0.11707   1.792   0.0734 .
## ResearchGroup1:time_visit -0.06842    0.32970  -0.208   0.8356
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 12.58436)
##
##     Null deviance: 21049  on 1205  degrees of freedom
## Residual deviance: 15076  on 1198  degrees of freedom
##   (558 observations deleted due to missingness)
## AIC: 6486.6
##
## Number of Fisher Scoring iterations: 2
```

```
#LMM
model.lmm<-lmer(UPDRS~Imaging+ResearchGroup*time_visit+Age+Sex+Weight+(time_
visit|FID_IID), data=mydata)
summary(model.lmm)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula:
## UPDRS ~ Imaging + ResearchGroup * time_visit + Age + Sex + Weight +
##     (time_visit | FID_IID)
##    Data: mydata
##
## REML criterion at convergence: 5737.9
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.2660 -0.4617 -0.0669  0.3575  4.6158
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  FID_IID  (Intercept) 7.8821   2.8075
##           time_visit  0.2454   0.4954   0.16
##  Residual             3.1233   1.7673
## Number of obs: 1206, groups:  FID_IID, 440
##
## Fixed effects:
```

```
##                             Estimate Std. Error t value
## (Intercept)                  0.69803    0.16881   4.135
## Imaging                      0.04200    0.02669   1.574
## ResearchGroup1              -6.93136    0.34425 -20.135
## time_visit                   0.02799    0.06385   0.438
## Age                          0.47720    0.15065   3.168
## Sex                          0.18662    0.17212   1.084
## Weight                       0.24146    0.17075   1.414
## ResearchGroup1:time_visit -0.04785    0.30496  -0.157
##
## Correlation of Fixed Effects:
##             (Intr) Imagng RsrcG1 tm_vst Age    Sex    Weight
## Imaging     -0.059
## ReserchGrp1 -0.496  0.101
## time_visit   0.067 -0.002 -0.033
## Age         -0.028  0.128  0.045  0.002
## Sex         -0.029  0.014  0.048  0.006  0.140
## Weight      -0.015  0.046  0.022  0.006  0.125  0.522
## RsrchGrp1:_ -0.011 -0.053 -0.001 -0.209 -0.010 -0.005  0.000
```

```
display(model.lmm)
```

```
## lmer(formula = UPDRS ~ Imaging + ResearchGroup * time_visit +
##     Age + Sex + Weight + (time_visit | FID_IID), data = mydata)
##                             coef.est coef.se
## (Intercept)                  0.70     0.17
## Imaging                      0.04     0.03
## ResearchGroup1              -6.93     0.34
## time_visit                   0.03     0.06
## Age                          0.48     0.15
## Sex                          0.19     0.17
## Weight                       0.24     0.17
## ResearchGroup1:time_visit -0.05     0.30
##
## Error terms:
##  Groups   Name        Std.Dev. Corr
##  FID_IID  (Intercept) 2.81
##           time_visit  0.50     0.16
##  Residual             1.77
## ---
## number of obs: 1206, groups: FID_IID, 440
## AIC = 5761.9, DIC = 5702.5
## deviance = 5720.2
```

Note that we use the notation `ResearchGroup*time_visit` that is identical to `ResearchGroup + time_visit + ResearchGroup*time_visit`. Here R will include both terms and their interaction into the model. According to the model outputs, the LMM model has a relatively smaller AIC. In terms of AIC, LMM may represent a better model fit than GLM.

19.3.2 Modeling the Correlation

In the summary of the LMM model, we can see a section called `Correlation of Fixed Effects`. The original model made no assumption about the correlation (unstructured correlation). In R, we usually have the following 4 types of correlation models.

- **Independence**: No correlation:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

- **Exchangeable**: Correlations are constant across measurements:

$$\begin{pmatrix} 1 & \rho & \rho \\ \rho & 1 & \rho \\ \rho & \rho & 1 \end{pmatrix}.$$

- **Autoregressive order 1(AR(1))**: Correlations are stronger for closer measurements and weaker for more distanced measurements:

$$\begin{pmatrix} 1 & \rho & \rho^2 \\ \rho & 1 & \rho \\ \rho^2 & \rho & 1 \end{pmatrix}.$$

- **Unstructured**: Correlation is different for each occasion:

$$\begin{pmatrix} 1 & \rho_{1,2} & \rho_{1,3} \\ \rho_{1,2} & 1 & \rho_{2,3} \\ \rho_{1,3} & \rho_{2,3} & 1 \end{pmatrix}.$$

In the LMM model, the output also seems unstructured. So, we needn't worry about changing the correlation structure. However, if the output under unstructured correlation assumption looks like an Exchangeable or AR(1) structure, we may consider changing the LMM correlation structure accordingly.

19.4 GLMM/GEE Longitudinal Data Analysis

If the response is a binary variable like `ResearchGroup`, we need to use General Linear Mixed Model (GLMM) instead of LMM. The marginal model of GLMM is called GEE. However, GLMM and GEE are actually different.

In situations where the responses are discrete, there may not be a uniform or systematic strategy for dealing with the joint multivariate distribution of $Y_i = \{(Y_{i1}, Y_{i2}, \ldots, Y_{in})\}^T$, . That's where the GEE method comes into play as it's based on the concept of estimating equations. It provides a general approach for analyzing discrete and continuous responses with marginal models.

GEE is applicable when:

1. β, a generalized linear model regression parameter, characterizes systematic variation across covariate levels,

2. The data represents repeated measurements, clustered data, multivariate response, and
3. The correlation structure is a nuisance feature of the data.

Notation

- Response variables: $\{Y_{i,1}, Y_{i,2}, \ldots, Y_{i,n_t}\}$, where $i \in [1, N]$ is the index for clusters or subjects, and $j \in [1, n_t]$ is the index of the measurement within cluster/subject.
- Covariate vector: $\{X_{i,1}, X_{i,2}, \ldots, X_{i,n_t}\}$.

The primary focus of GEE is the estimation of the **mean model**: $E(Y_{i,\,j} | X_{i,\,j}) = \mu_{i,\,j}$, where

$$g(\mu_{i,j}) = \beta_0 + \beta_1 X_{i,j}(1) + \beta_2 X_{i,j}(2) + \beta_3 X_{i,j}(3) + \ldots + \beta_p X_{i,j}(p) = X_{i,j} \times \beta.$$

This mean model can be any generalized linear model. For example: $P(Y_{i,\,j} = 1 | X_{i,\,j}) = \pi_{i,\,j}$ (marginal probability, as we don't condition on any other variables):

$$g(\mu_{i,j}) = \ln\left(\frac{\pi_{i,j}}{1 - \pi_{i,j}}\right) = X_{i,j} \times \beta.$$

Since the data could be clustered (e.g., within subject, or within unit), we need to choose a correlation model. Let's introduce some notation:

$$V_{i,j} = var(Y_{i,j}|X_i),$$
$$A_i = diag(V_{i,j}),$$

the paired correlations:

$$\rho_{i,j,k} = corr(Y_{i,j}, Y_{i,k}|X_i),$$

the correlation matrix:

$$R_i = (\rho_{i,j,k}), \text{for all } j \text{ and } k,$$

and the paired predictor-response covariances are:

$$V_i = cov(Y_i|X_i) = A_i^{1/2} R_i A_i^{1/2}.$$

Assuming different correlation structures in the data leads to alternative models, see the examples above.

Notes

- GEE is a semi-parametric technique because:
  - The specification of a mean model, $\mu_{i,\,j}(\beta)$, and a correlation model, $R_i(\alpha)$, does not identify a complete probability model for Y_i

  - The model $\{\mu_{i,\ j}(\beta), R_i(\alpha)\}$ is semi-parametric since it only specifies the first two multivariate moments (mean and covariance) of Y_i. Higher order moments are not specified.

- Without an explicit likelihood function, to estimate the parameter vector β (and perhaps the covariance parameter matrix $R_i(\alpha)$) and perform a valid statistical inference that takes the dependence into consideration, we need to construct an unbiased estimating function:
- $D_i(\beta) = \frac{\partial \mu_i}{\partial \beta}$, the partial derivative, w.r.t. β, of the mean-model for subject i.
- $D_i(j,k) = \frac{\partial \mu_{i,j}}{\partial \beta_k}$, the partial derivative, w.r.t. β, , the partial derivative, w.r.t. the kth regression coefficient (βk), of the mean-model for subject i and measurement (e.g., time-point) j.

  Estimating (cost) function:

$$U(\beta) = \sum_{i=1}^{N} D_i^T(\beta) V_i^{-1}(\beta, \alpha)\{Y_i - \mu_i(\beta)\}.$$

  Solving the Estimating Equations leads to parameter estimating solutions:

$$0 = U\left(\hat{\beta}\right) = \sum_{i=1}^{N} \underbrace{D_i^T\left(\hat{\beta}\right)}_{\text{scale}} \underbrace{\left(V_i^{-1}\hat{\beta}, \alpha\right)}_{\text{variance weight}} \underbrace{\left\{Y_i - \mu_i\left(\hat{\beta}\right)\right\}}_{\text{model mean}}.$$

Scale: A change of scale term transforming the scale of the mean, μ_i, to the scale of the regression coefficients (covariates).

Variance weight: The inverse of the variance-covariance matrix is used to weight in the data for subject i, i.e., giving more weight to differences between observed and expected values for subjects that contribute more information.

Model Mean: Specifies the mean model, $\mu_i(\beta)$, compared to the observed data, Y_i. This fidelity term minimizes the difference between actually-observed and mean-expected (within the ith cluster/subject). See also the SMHS EBook.

19.4.1 GEE Versus GLMM

There is a difference in the interpretation of the model coefficients between GEE and GLMM. The fundamental difference between GEE and GLMM is in the target of the inference: population-average vs. subject-specific. For instance, consider an example where the observations are dichotomous outcomes (Y), e.g., single Bernoulli trials or death/survival of a clinical procedure, that are grouped/clustered into hospitals and units within hospitals, with N additional demographic, phenotypic, imaging and genetics predictors. To model the failure rate between genders (males vs. females) in a hospital, where all patients are spread among different hospital units (or clinical teams), let Y represent the binary response (death or survival).

In GLMM, the model will be pretty similar with the LMM model.

$$log\left(\frac{P(Y_{ij}=1)}{P(Y_{ij}=0)}|X_{ij},b_i\right)=\beta_0+\beta_1 x_{ij}+b_i+\epsilon_{ij}.$$

The only difference between GLMM and LMM in this situation is that GLMM used a *logit link* for the binary response.

With GEE, we don't have random intercept or slope terms.

$$log\left(\frac{P(Y_{ij}=1)}{P(Y_{ij}=0)}|X_{ij},b_i\right)=\beta_0+\beta_1 x_{ij}+\epsilon_{ij}.$$

In the marginal model (GEE), we are ignoring differences among hospital-units and just aim to obtain population (hospital-wise) rates of failure (patient death) and its association with patient gender. The GEE model fit estimates the odds ratio representing the population-averaged (hospital-wide) odds of failure associated with patient gender.

Thus, parameter estimates ($\hat{\beta}$) from GEE and GLMM models may differ because they estimate different things.

Let's compare the results of the GLM and GLMM models for our PPMI dataset.

```
# install.packages("gee")
library(gee)
model.glm1<-glm(ResearchGroup~UPDRS+Imaging+Age+Sex+Weight, data = mydata, f
amily="binomial")

model.glm1

##
## Call:  glm(formula = ResearchGroup ~ UPDRS + Imaging + Age + Sex + Weight
,
##     family = "binomial", data = mydata)
##
## Coefficients:
## (Intercept)        UPDRS      Imaging          Age          Sex
##   -10.64144     -1.96707      0.03889      0.71562      0.19361
##      Weight
##     0.40606
##
## Degrees of Freedom: 1205 Total (i.e. Null);  1200 Residual
##   (558 observations deleted due to missingness)
## Null Deviance:       811.9
## Residual Deviance: 195.8     AIC: 207.8

#mydata1<-na.omit(mydata)
#attach(mydata1)
#model.gee<-gee(ResearchGroup~L_insular_cortex_ComputeArea+L_insular_cortex_
Volume+ Sex + Weight + Age + chr17_rs11012_GT + chr17_rs199533_GT + UPDRS_pa
rt_I + UPDRS_part_II + time_visit, id=FID_IID, data = mydata1, family=binomi
al(link = logit))
```

```
model.glmm<-glmer(ResearchGroup~UPDRS+Imaging+Age+Sex+Weight+(1|FID_IID), da
ta=mydata, family="binomial")
display(model.glmm)

## glmer(formula = ResearchGroup ~ UPDRS + Imaging + Age + Sex +
##     Weight + (1 | FID_IID), data = mydata, family = "binomial")
##             coef.est coef.se
## (Intercept) -86.63    32.07
## UPDRS       -16.78     6.27
## Imaging       0.59     0.61
## Age           6.04     2.41
## Sex           0.65     2.15
## Weight        6.12     3.76
##
## Error terms:
##  Groups   Name        Std.Dev.
##  FID_IID  (Intercept) 40.72
##  Residual              1.00
## ---
## number of obs: 1206, groups: FID_IID, 440
## AIC = 129.5, DIC = -114.1
## deviance = 0.7
```

In terms of AIC, the GLMM model is a lot better than the GLM model.

Try to apply some of these longitudinal data analytics on the fMRI data we discussed in Chap. 4 (Visualization).

19.5 Assignment: 19. Big Longitudinal Data Analysis

19.5.1 Imaging Data

Review the 3D/4D MRI imaging data discussion in Chap. 4. Extract the time courses of several time series at different 3D spatial locations, some near-by, and some farther apart (distant voxels). Then, apply time-series analyses, report findings, determine if near-by or farther-apart voxels may be more correlated.

Example of extracting time series from 4D fMRI data:

```
#See examples here: https://cran.r-project.org/web/packages/oro.nifti/vignettes/nifti.pdf

fMRIURL <- "http://socr.umich.edu/HTML5/BrainViewer/data/fMRI_FilteredData_4D.nii.gz"
fMRIFile <- file.path(tempdir(), "fMRI_FilteredData_4D.nii.gz")
download.file(fMRIURL, dest=fMRIFile, quiet=TRUE)
(fMRIVolume <- readNIfTI(fMRIFile, reorient=FALSE))
# dimensions: 64 x 64 x 21 x 180 ; 4mm x 4mm x 6mm x 3 sec

fMRIVolDims <- dim(fMRIVolume); fMRIVolDims
time_dim <- fMRIVolDims[4]; time_dim

hist(fMRIVolume)
```

```
  # To examine the time course of a specific 3D voxel (say the one at x=30, y=30, z=15):
  plot(fMRIVolume[30, 30, 10,], type='l', main="Time Series of 3D Voxel \n (x=30, y=30, z=1
5)", col="blue")

  x1 <- c(1:180)
  y1 <- loess(fMRIVolume[30, 30, 10,]~ x1, family = "gaussian")
  lines(x1, smooth(fMRIVolume[30, 30, 10,]), col = "red", lwd = 2)
  lines(ksmooth(x1, fMRIVolume[30, 30, 10,], kernel = "normal", bandwidth = 5), col = "gree
n", lwd = 3)
```

19.5.2 Time Series Analysis

Use Google Web-Search Trends and Stock Market Data to:

- Plot time series for the variable `Job`.
- Apply `TTR` to smooth the original graph by month.
- Determine the differencing parameter.
- Decide the auto-regressive (AR) and moving average (MA) parameters.
- Build an ARIMA model, forecast the `Job` variable over the next year and evaluate this model.

19.5.3 Latent Variables Model

Use the Hand written English Letters data to:

- Explore the data and evaluate the correlations between covariates.
- Justify the application of a latent variable model.
- Apply proper data conversion and scaling.
- Fit a Structural Equation Model (SEM) using the lavaan::cfa() function for these data by adding proper latent variable.
- Summarize and interpret the outputs.
- Use the model you found above to fit *GEE* and *GLMM* models using the latent variable as response and compare the models using AIC. (Hint: add a fake variable as random effect for GLMM).

References

Box GE, Jenkins GM, Reinsel GC, Ljung GM. Time series analysis: forecasting and control: John Wiley & Sons; 2015.

Grace JB. Structural equation modeling and natural systems: Cambridge University Press; 2006.

http://idaejin.github.io/bcam-courses/neiker-2016/material/mixed-models/

Liang K-Y, Zeger S. Longitudinal data analysis using generalized linear models. Biometrika. 1986;73(1):13-22. doi: https://doi.org/10.1093/biomet/73.1.13.

McCulloch CE, Neuhaus JM. Generalized linear mixed models: Wiley Online Library; 2013.

McIntosh A, Gonzalez-Lima F. Structural equation modeling and its application to network analysis in functional brain imaging. Human Brain Mapping. 1994;2(1-2):2-22.

Shipley B. Cause and correlation in biology: a user's guide to path analysis, structural equations and causal inference with R: Cambridge University Press; 2016.