リトルブック：R による時系列解析

A Little Book of R For Time Series, Release 0.1

2011 年 8 月 29 日

Avril Coghlan

日本語訳　荒木　孝治

2012 年 2 月 18 日

著者：Avril Coghlan（University College Cork, Cork, Ireland）．E メール：a.coghlan@ucc.ie

本ブックレットは，R を用いた時系列解析への簡単な入門書である[*1]．

本ブックレットの pdf 版は，https://github.com/avrilcoghlan/LittleBookofRTimeSeries/raw/master/_build/latex/TimeSeries.pdf より取得可能である．

本ブックレットが気に入った人は，“R による生物医学統計学”（http://alittle-book-of-r-for-biomedical-statistics.readthedocs.org/）と “R による多変量解析”（http://little-book-of-r-for-multivariate-analysis.readthedocs.org/）も参照してほしい．

[*1] 本翻訳に関するコメント・問い合わせ等は，荒木孝治（関西大学商学部，arakit@kansai-u.ac.jp）までお願いします．

目次

第 1 章

R のインストール方法

1.1 R とは

本ブックレットは，生物医学統計で R を利用する方法に関する入門書である．

R（www.r-project.org）は，一般的に用いられるフリーの統計ソフトウェアである．R は，対話モードのみならず簡単なプログラミングにより統計分析を実行することができる．

1.2 R のインストール

R を使うには，R プログラムがコンピュータにインストールされている必要がある．

1.2.1 R が Windows パソコンにインストールされているかどうかを確認する方法

R をコンピュータにインストールする前にすべきことは，R が，例えば前のユーザーによって既にインストールされているかどうかを調べることである．

以下，主に Windows パソコンに関してその方法について説明するが，Macintosh または Linux コンピュータに関しても少し触れる．Windows パソコンの場合，R がコンピュータに既にインストールされているかどうか調べる方法には, 次に示す 2 つがある．

1. “R” のアイコンがコンピュータのデスクトップにあるかどうかを調べる．もしあれば，“R” アイコンをダブルクリックすることにより，R を起動することができる．無いときは，手順 2 に行く．
2. Windows のデスクトップの左下にある “スタート” ボタンをクリックし，ポップアップしたメニューの［全てのプログラム］の上にマウスを移動させる．表示されるプログラムのリストに “R” があるかどうかを確認する．もしあれば，すでに R がインストールされているので，リストから “R”（例えば，R 2.10.0（R X.X.X の X.X.X は R のバージョンを意味する））を選択することによって R をスタートさせることができる．

上記の (1) か (2) により R を起動できたら，既にコンピュータに R がインストールされることを意味する（どちらもだめなときは，R はまだインストールされていない）．インストールされている R のバージョンが古い場合，最新版をインストールする方がよい．最新の R の機能を利用することができるからである．

1.2.2 R の最新版を知る方法

R の最新バージョンは，CRAN（The Comprehensive R Archive Network）http://cran.r-project.org/ で確認することができる．

“The latest release”（最新のリリース）の他にも（ページの途中に），“R-X.X.X.tar.gz”（例えば，

“R-2.12.1.tar.gz”）のような表示がある．これは，R の最新のバージョンが X.X.X（今の場合，2.12.1）であることを意味する．

R の開発は非常に活発なため，R の新しいバージョンのリリースは定期的に行われている（年におよそ 2 回）．定期的に R の新しいバージョンを確認し，それをインストールすることには価値がある（それにより，ダウンロードした R のパッケージのすべての最新版との互換性を確実にすることができる）．

1.2.3 Windows パソコンへの R のインストール

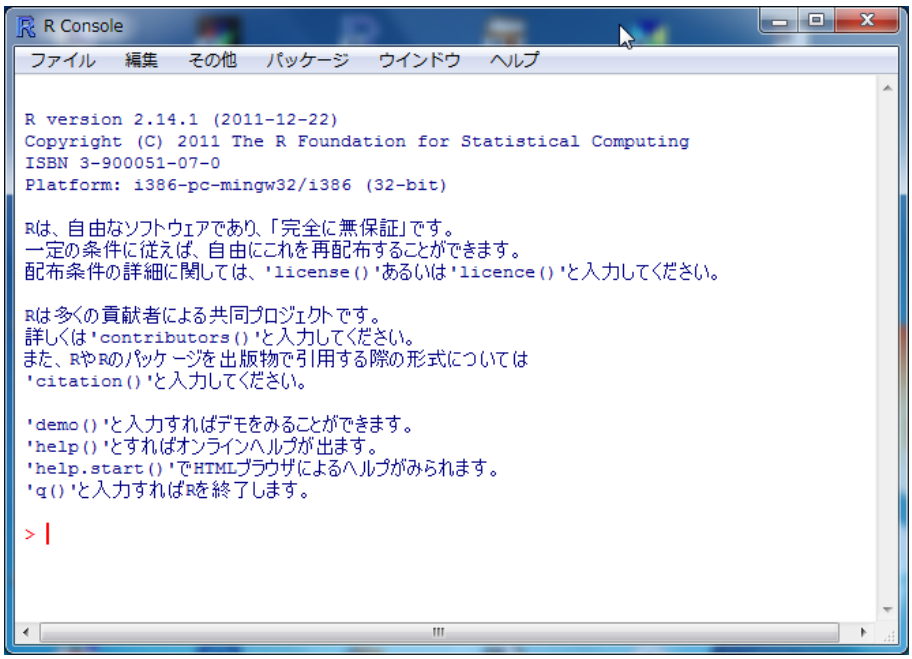
Windows パソコンに R をインストールするには，次の手順を実行する．

1. `http://essrc.hyogo-u.ac.jp/cran/` へ行く[*1]
2. “Download R for Windows” をクリック．
3. “Subdirectories” で “base” をクリック．
4. 次のページに,“Download R 2.12.1 for Windows” (R X.X.X．の “X.X.X” は R のバージョンを示す．例えば，R 2.12.1)[*2]といった表示がある． このリンクをクリック．
5. ファイル “R-2.12.1-win32.exe” をどう処理するか，つまり，保存するか開く（実行する）かを聞かれるので，“ファイルを保存する” を選択し，保存場所をデスクトップに指定する．保存後，ファイルをダブルクリックしてインストールを実行する．
6. インストール中に利用する言語を聞かれるので，英語（English）を選択する[*3]．
7. R のセットアップウイザードが起動するので，下にある “Next”（“次へ”）ボタンをクリックする．
8. “Information”（“情報”）ページが開かれる．“Next”（“次へ”）をクリック．
9. “Select Destination Location”（“インストール先の指定”）ページが開かれる．デフォルトのインストール場所は “C:¥Program Files” である．
10. セットアップウイザードの下にある “Next”（“次へ”）をクリックする．
11. “Select Components”（“コンポーネントの選択”）ページが開かれる．“Next”（“次へ”）をクリック．
12. “Startup options” ページが開かれる．“‘Next”（“次へ”）をクリック．
13. “Select Start Menu Folder” ページが開かれる．“Next” をクリック．
14. “Select Additional Tasks”（“追加タスクの選択”）ページが開かれる．“Next”（“次へ”）クリック．
15. これで R がインストールされる．インストールには 1 分くらいかかる．終了すると “Completing the R for Windows Setup Wizard”（“セットアップウィザードの完了”）が表示される．“Finish”（“完了”）をクリックする．
16. R を起動するには，手順 18 または 19 を実行する．
17. “R” のアイコンがコンピュータのデスクトップに表示されているかどうかを確認する．表示されている場合，R を起動するには “R” をダブルクリックする．表示されていない場合，手順 19 を実行する．
18. コンピュータ・スクリーンの左下にある “スタート” ボタンをクリックし，“すべてのプログラム” を選択する．次にプログラムのメニューから “R”（R X.X.X の “X.X.X” は R のバージョンを示し，例えば，R 2.12.1）を選択することにより，R を起動する．
19. R コンソールが表示される．

[*1] （訳注）日本のミラーサイトに変更．原著では，`http://ftp.heanet.ie/mirrors/cran.r-project.org`.

[*2] （訳注）原著では，バージョンが様々なので，表記を “R 2.12.1” に統一した（以下，同様）．

[*3] （訳注）Japanese（日本語）でのインストールを選択できるが，一部文字化けするので，英語でインストールする方がよい．



1.2.4 非 Windows オペレーティングシステム（例えば，Macintosh，または Linux）へのインストール

上記は，Windows パソコンへの R のインストール方法である．非 Windows オペレーティングシステム（OS）を利用するコンピュータ（例えば，Macintosh または Linux）の場合，`http://ftp.heanet.ie/mirrors/cran.r-project.org` から，OS に適切な R インストーラをダウンロードし，`http://ftp.heanet.ie/mirrors/cran.rproject` にある R のインストール方法に従う．

1.3 R のパッケージのインストール

R をインストールするとき，いくつかの標準パッケージも一緒にインストールされる．有用な他の R パッケージ（たとえば，“rmeta” パッケージ）を使う方法を本ブックレットでは説明する．これらの付加的なパッケージは R と一緒にインストールされないので，別にインストールする必要がある．

1.3.1 R のパッケージのインストール法

R を Windows パソコンに（上記のステップに従って）インストールした後，下記の手順でパッケージを追加的にインストールすることができる．

1. R を起動するために，次の手順 2 または 3 を実行する．
2. “R” のアイコンがデスクトップにあるかどうか調べる．あるなら，“R” アイコンをダブルクリックして R を起動する．ない場合，手順 3 を実行する．
3. コンピュータ・スクリーンの左下にある “スタート” ボタンをクリックし，“すべてのプログラム” から “R”（R X.X.X の X.X.X は R のバージョンで，例えば，R 2.12.1）を選択することにより，R を起動する．
4. R コンソールが表示される．
5. R を起動すると，R コンソールの上にある “パッケージのインストール” メニューより R のパッケージ（例えば，“rmeta”）をインストールすることができる．どのウェブサイトからダウンロードするかを聞かれるので，“Japan”（または他の国）を選択する．インストール可能なパッケージのリストが表示されるので，そのリストからインストールしたいパッケージ（例えば，

“rmeta”）を指定する．

6. これにより “rmeta” パッケージのインストールが始まる．
7. “rmeta” パッケージのインストールが終了する．この後，R を起動した後，R コンソールに次のコマンドを入力することにより “rmeta” パッケージをロードし，利用することができる．

```
> library("rmeta")
```

Bioconductor（http://www.bioconductor.org）というバイオインフォマティックスのための R パッケージの特殊な集合がある（例えば，“yeastExpData” や “Biostrings” といったパッケージの集まり）．Bioconductor パッケージ群は，Bioconductor に特有の手順（Bioconductor の R パッケージをインストールする方法を参照）によりインストールする必要がある．

1.3.2 Bioconductor のパッケージのインストール法

1.3.1 項に示した手順で，R パッケージの大部分をインストールすることができる．しかし，バイオインフォマティックス用の R パッケージの集合である Bioconductor に関しては，特別な手順に従う必要がある．Bioconductor（http://www.bioconductor.org）は，バイオインフォマティックスのために開発されたパッケージ群である．

1. R を起動するには，次の手順 2 または 3 を実行する．
2. “R” のアイコンがデスクトップにあるかどうか調べる．あるなら，“R” アイコンをダブルクリックして R を起動する．“R” アイコンがない場合，手順 3 を実行する．
3. コンピュータ・スクリーンの左下にある “スタート” ボタンをクリックし，“すべてのプログラム” から “R”（R X.X.X の X.X.X は R のバージョンで，例えば，R 2.12.1）を選択することにより，R を起動する．
4. R コンソールが表示される．
5. R を起動した後，R コンソールに次を入力する．

```
> source("http://bioconductor.org/biocLite.R")
> biocLite()
```

6. これにより Bioconductor の主要パッケージ（“affy”，“affydata”，“affyPLM”，“annaffy”，“annotate”，“Biobase”，“Biostrings”，“DynDoc”，“gcrma”，“genefilter”，“geneplotter”，“hgu95av2.db”，“limma”，“marray”，“matchprobes”，“multtest”，“ROC”，“vsn”，“xtable”，“affyQCReport”）がインストールされる．これには少し時間がかかる．
7. 後日，Bioconductor の主要パッケージ以外のパッケージ，例えば，“yeastExpData” をインストールする必要が生じたときは，R コンソールに次を入力する．

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("yeastExpData")
```

8. パッケージをインストールした後，それを利用するには R コンソールに次を入力する．

```
> library("yeastExpData")
```

1.4 R の起動

R を利用するには，先ず R プログラムを起動する．それには，R がインストールされている必要がある．R を起動するには次の手順 1 または 2 を実行する．

1. “R” のアイコンがデスクトップにあるかどうか調べる．あるなら，“R” アイコンをダブルクリッ

クして R を起動する．“R” アイコンがない場合，手順 2 を実行する．

2. コンピュータ・スクリーンの左下にある “スタート” ボタンをクリックし，“すべてのプログラム” から “R”（R X.X.X の “X.X.X” は R のバージョンで，例えば，R 2.12.1）を選択することにより，R を起動する．

これにより，R コンソールという新しいウインドウが表示される．

1.5 R 超入門

R 内で分析を実行するには，R コンソールにコマンドを入力する．R コンソールには次に示す記号が表示されている．

```
>
```

この記号（>）は R のプロンプトである．プロンプトの後ろに，特定の作業をするのに必要なコマンドを入力する．コマンドは，Return キーを入力後に実行される．R をいったん起動すると，コマンドを入力することにより R を利用でき，結果はすぐに表示される．例えば：

```
> 2*3
[1] 6
> 10-3
[1] 7
```

R が生成した全ての変数（スカラー，ベクトル，行列等）はオブジェクトと呼ばれる．R では，変数に値を与えるとき，矢印（<-）を用いる．例えば，値 2*3 を変数 x に与えるには次のようにする．

```
> x <- 2 * 3
```

R のオブジェクトの内容を見るには，その名前を入力するだけでよい．その内容が表示される．

```
> x
[1] 6
```

R には，スカラー，ベクトル，行列，配列，データフレーム，テーブル，リストといった様々な種類のオブジェクトがある．上記のスカラー変数 x は，R オブジェクトの 1 例である．スカラー変数はちょうど 1 つの要素を持つが，ベクトルはいくつかの要素から構成される．c() 関数（c は combine の c）を用いて，ベクトルを作成することができる．例えば値 8，6，9，10，5 の要素から構成される，*myvector* という名前のベクトルを作成するには，次を入力する．

```
> myvector <- c(8, 6, 9, 10, 5)
```

変数 *myvector* の内容を表示するには，その名前を入力するだけでよい．

```
> myvector
[1] 8 6 9 10 5
```

[1] はベクトルの 1 番目の要素を示すインデックスである．ベクトルの任意の要素を抽出するには，角括弧（[ ]）の中に要素のインデックスを与えたものをベクトルの名前の後ろに記載する．例えば，ベクトル *myvector* の 4 番目の要素の値を抽出するには，次のようにする．

```
> myvector[4]
[1] 10
```

リストは，ベクトルと異なり，数値と記号といった異なる型の要素を保持することができる．リストはまた，ベクトルなどの他の変数を含むこともできる．リストの作成には list() 関数を用いる．

例えば，*mylist* というリストを作るには，次のコマンドを入力する．

```
> mylist <- list(name="Fred", wife="Mary", myvector)
```

リスト *mylist* の内容を表示するには，その名前を入力する．

```
> mylist
$name
[1] "Fred"

$wife
[1] "Mary"

[[3]]
[1]  8  6  9 10  5
```

リストの要素には番号が付けられており，それをインデックスとして参照することができる．リストの要素を抽出するには，リスト名の後ろに，2 重の角括弧（[[ ]]）の中に要素のインデックスを与えたものを記載することにより可能である．だから，2 番目と 3 番目の要素を *mylist* から抽出するには次のようにする．

```
> mylist[[2]]
[1] "Mary"
> mylist[[3]]
[1] 8 6 9 10 5
```

リストの要素に名前を付けることができる．この場合，名前の後ろに “$” をつけ，続いて要素名を記載することでリストの要素を参照することができる．例えば，*mylist*$*name* は *mylist*[[1]] と同じであり，*mylist*$*wife* は *mylist*[[2]] と同じである．

```
> mylist$wife
[1] "Mary"
```

リスト中の名前が与えられた要素の名前を知るには，attributes() 関数を用いて次のようにする．

```
> attributes(mylist)
$names
[1] "name" "wife" ""
```

リスト変数が持つ名前付きの要素を知るために attributes() 関数を用いると，名前付き要素には常に，“$names” というヘッダーが付けられる．よって，リスト変数 *mylist* の名前付き要素の名前が “*name*” と “*wife*” であることがわかる．*mylist*$*name* や *mylist*$*wife* と入力することによってそれらの値を切り出すことができる．

R で利用する別のオブジェクトとして，テーブル変数がある．例えば，学級内の生徒の名前を含む名前のベクトル変数 *mynames* があるとする．table() 関数を利用して，*mynames* 内の同じ名前を持つ子供の数のテーブル変数を作るには次のようにする．

```
> mynames <- c("Mary", "John", "Ann", "Sinead", "Joe", "Mary", "Jim", "John", "Simon")
> table(mynames)
mynames
   Ann    Jim    Joe   John   Mary  Simon Sinead
     1      1      1      2      2      1      1
```

関数 table() によって作ったテーブル変数に，名前 “*mytable*” をつけて保存するには次を入力する．

```
> mytable <- table(mynames)
```

テーブル変数の要素にアクセスするには，リストの要素にアクセスするときと同様に，二重の角括弧（[[ ]]）を使う．例えば，*mytable* の 4 番目の要素（“John” という子供の数）にアクセスするには，

次を入力する．

```
> mytable[[4]]
[1] 2
```

表の 4 番目の要素の名前（“John”）を用いてテーブル要素の値を知ることもできる．

```
> mytable[["John"]]
[1] 2
```

R の関数は通常，引数を必要とする．引数は，関数に渡される入力変数（オブジェクト）であり，それは，演算を実行するときに利用される．例えば，log10() 関数は数を渡され，その数の，底が 10 の対数の値を計算する．

```
> log10(100)
[1] 2
```

R では，help() 関数を用いて特定の関数についてヘルプ画面を参照することができる．例えば，log10() 関数についてヘルプを参照したいとき，次を入力する．

```
> help("log10")
```

help() 関数を利用するとき，ウィンドウまたはウェブページが出現し，ヘルプを求める関数に関する情報が表示される．

関数の名前に確信がないが，その名前の一部がわかっているとき，help.search() と RSiteSearch() 関数を使って関数名を探すことができる．help.search() 関数は，興味があるトピックに関連する関数がすでにインストールされて（インストールされているパッケージのどれかに入って）いるかどうかを探す．RSiteSearch() 関数は，興味があるトピックに関連した関数を，すべての R 関数（まだインストールされていないパッケージに含まれるものを含んで）の中から探す．

例えば，標準偏差を計算する関数があるかどうかを知りたいとき，次を入力することにより，関数に関する記述の中から “deviation”（偏差）という語を含むすべてのインストールされた関数を探すことができる．

```
> help.search("deviation")
Help files with alias or concept or title matching
'deviation' using fuzzy matching:

genefilter::rowSds
                    Row variance and standard deviation of
                    a numeric array
nlme::pooledSD      Extract Pooled Standard Deviation
stats::mad          Median Absolute Deviation
stats::sd           Standard Deviation
vsn::meanSdPlot     Plot row standard deviations versus row
```

見つけた関数の中に，“stats” パッケージ（R のインストールとともにインストールされる基本パッケージ）の中に sd() 関数があり，これにより標準偏差を計算することができることがわかる．

上記の例では，help.search() 関数は関連した関数（sd()）を見つけた．help.search() で期待していたものを見つけることができない場合，RSiteSearch() 関数を使うことにより，R のウェブサイトで記述されているすべての関数の中から，興味があるトピックに関連するものを見つけることができるかもしれない．

```
> RSiteSearch("deviation")
```

RSiteSearch() 関数の結果は，R 関数の記述への，ならびにそれらの関数に関する R メーリングリスト議論へのヒット結果である．

R では，スカラーとベクトルといったオブジェクトを利用して計算することができる．例えば，ベクトル myvector の値の平均を計算する（すなわち，8，6，9，10，5 の平均）ために，mean() 関数を使うことができる．

```
> mean(myvector)
[1] 7.6
```

mean()，length()，print()，plot() といった R の組み込み関数を利用することができる．これに対し，よく利用する機能を持つ関数を独自に作成することもできる．例えば，入力された数の 2 乗に 20 を加えるための関数を作成するには次のようにする．

```
> myfunction <- function(x) { return(20 + (x*x)) }
```

return() 関数は，計算値を返す機能を持つ．この関数を入力した後，関数を使うことができる．例えば，異なる入力値（例えば，10，25）に対してこの関数を使うことができる．

```
> myfunction(10)
[1] 120
> myfunction(25)
[1] 645
```

R を終了するには次を入力する．

```
> q()
```

1.6 リンクと参考文献

さらに学習するためのリンクを紹介する．

R へのより詳細に入門するための良いオンライン・チュートリアルが，“Kickstarting R” というウェブサイト（cran.rproject.org/doc/contrib/Lemon-kickstart）にある．

別の素晴らしい，もう少し詳細なチュートリアルが，“Introduction to R” というウェブサイト（cran.rproject.org/doc/manuals/R-intro.html）にある．

1.7 謝辞

Friedrich Leisch と Phil Spector には，R のインストールに記述に関して非常に有益なコメントと提案をいただいた．感謝する．

1.8 連絡

訂正や改良に関する提案があれば，著者（Avril Coghlan）のメールアドレス（a.coghlan@ucc.ie）まで送付していただければ幸いである．

1.9 ライセンス

第 2 章

R による時系列解析

2.1 時系列解析

このブックレットは、時系列データを分析する際によく用いられる簡単な手法を R を用いて実行する方法を示す．読者は，時系列解析の入門的な知識を持つことを前提とする．このブックレットの主目的は，時系列解析自体を説明することにはなく，R を用いてその手法を実践する方法を説明することにある．

時系列解析を知らなかったり，このブックレットで説明する概念についてもっと知りたかったりする人には，Open University 刊行の本 “Time series“（製品コード $M249/02$）を参照することを強く勧める（Open University のショップで購入可）．

このブックレットでは，Rob Hyndman が時系列データライブラリ（Time Series Data Library : http://robjhyndman.com/TSDL/）より提供してくれた時系列データセットを利用する．

このブックレットは, ウェブサイト https://github.com/avrilcoghlan/LittleBookofRTimeSeries/raw/master/_build/latex/TimeSeries の pdf 版である．

このブックレットを気に入った人は, “R を用いた生物医療統計”（http://alittle-book-of-r-for-biomedical-statistics.readthedocs.org/）や “R を用いた多変量解析”（http://little-book-of-r-for-multivariate-analysis.readthedocs.org/）というブックレットも参照してほしい．

2.2 時系列データの読み込み

時系列データを分析する際に必要なことは，先ずそれを R に読み込み，図に描くことである．scan() 関数を用いて R にデータを読み込むことができるが，それには，連続した時点のデータがテキストファイルの中に列として入力されている必要がある．

たとえば，ファイル http://robjhyndman.com/tsdldata/misc/kings.dat にある “*kings.dat*” というデータセットは，征服王ウイリアム（William the Conqueror）からスタートするイングランドの歴代の王の享年のデータである．

データセットの一部を次に示す．

```
Age of Death of Successive Kings of England
#starting with William the Conqueror
#Source: McNeill, "Interactive Data Analysis"
60
43
67
50
56
42
```

```
50
65
13
68
43
65
34
...
```

最初の数行のみを表示している．最初の 3 行はデータに関するコメントなので，R に読み込むときはこれらを無視したい．それには，scan() 関数のパラメータとして “skip” を用いることにより可能である．これにより，ファイルの最初の何行を無視するかを指定することができる．最初の 3 行を無視して R にファイルを読み込むには，次を入力する．

```
> kings <- scan("http://robjhyndman.com/tsdldata/misc/kings.dat",skip=3)
Read 42 items
> kings
 [1] 60 43 67 50 56 42 50 65 68 43 65 34 47 34 49 41 13 35 53 56 16 43 69 59 48
[26] 59 86 55 68 51 33 49 67 77 81 67 71 81 68 70 77 56
```

これにより，イングランドの歴代 42 名の王の享年のデータが “*kings*” という変数に読み込まれた．

時系列データセットを読み込んだ後，次に行うべきことはそれを時系列オブジェクトとして保存することである．これにより，時系列データを分析するためのたくさんの R の関数を利用することができる．データを時系列オブジェクトとして保存するには ts() 関数を利用する．例えば，変数 “*kings*” を時系列オブジェクトとして R に保存するには，次のようにする．

```
> kingstimeseries <- ts(kings)
> kingstimeseries
Time Series:
Start = 1
End = 42
Frequency = 1
 [1] 60 43 67 50 56 42 50 65 68 43 65 34 47 34 49 41 13 35 53 56 16 43 69 59 48
[26] 59 86 55 68 51 33 49 67 77 81 67 71 81 68 70 77 56
```

時系列データには，一年未満の一定間隔（例えば，月次または四半期）で集められたものがある．この場合，ts() 関数の中で ‘frequency’ パラメータを用いて，年当たり何回観測したかを指定することができる．月次に対しては，frequency=12 とし，四半期に対しては，frequency=4 とする．

また，ts() 関数の中で，データを収集した最初の年，およびその年における最初のインターバルを ‘start’ パラメータにより指定することができる．

例として，1946 年 1 月から 1959 年 12 月までのニューヨークにおける出生数のデータセット（オリジナルのデータは Newton による）を取り上げる．次のようにして，このデータを http://robjhyndman.com/tsdldata/data/nybirths.dat にあるファイルから読み込み，時系列オブジェクトとして保存することができる．

```
> births <- scan("http://robjhyndman.com/tsdldata/data/nybirths.dat")
Read 168 items
> birthstimeseries <- ts(births, frequency=12, start=c(1946,1))
> birthstimeseries
        Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct    Nov    Dec
1946 26.663 23.598 26.931 24.740 25.806 24.364 24.477 23.901 23.175 23.227 21.672 21.870
1947 21.439 21.089 23.709 21.669 21.752 20.761 23.479 23.824 23.105 23.110 21.759 22.073
1948 21.937 20.035 23.590 21.672 22.222 22.123 23.950 23.504 22.238 23.142 21.059 21.573
1949 21.548 20.000 22.424 20.615 21.761 22.874 24.104 23.748 23.262 22.907 21.519 22.025
```

```
1950 22.604 20.894 24.677 23.673 25.320 23.583 24.671 24.454 24.122 24.252 22.084 22.991
1951 23.287 23.049 25.076 24.037 24.430 24.667 26.451 25.618 25.014 25.110 22.964 23.981
1952 23.798 22.270 24.775 22.646 23.988 24.737 26.276 25.816 25.210 25.199 23.162 24.707
1953 24.364 22.644 25.565 24.062 25.431 24.635 27.009 26.606 26.268 26.462 25.246 25.180
1954 24.657 23.304 26.982 26.199 27.210 26.122 26.706 26.878 26.152 26.379 24.712 25.688
1955 24.990 24.239 26.721 23.475 24.767 26.219 28.361 28.599 27.914 27.784 25.693 26.881
1956 26.217 24.218 27.914 26.975 28.527 27.139 28.982 28.169 28.056 29.136 26.291 26.987
1957 26.589 24.848 27.543 26.896 28.878 27.390 28.065 28.141 29.048 28.484 26.634 27.735
1958 27.132 24.924 28.963 26.589 27.931 28.009 29.229 28.759 28.405 27.945 25.912 26.619
1959 26.076 25.286 27.660 25.951 26.398 25.565 28.865 30.000 29.261 29.012 26.992 27.897
```

同様に，http://robjhyndman.com/tsdldata/data/fancy.dat のファイルには，1987 年 1 月から 1993 年 12 月までのオーストラリアのクイーンズランドのリゾートタウンの土産物店における月間売上高データがある（オリジナルデータは，Wheelwright and Hyndman, 1998，による）．

```
> souvenir <- scan("http://robjhyndman.com/tsdldata/data/fancy.dat")
Read 84 items
> souvenirtimeseries <- ts(souvenir, frequency=12, start=c(1987,1))
> souvenirtimeseries
          Jan       Feb       Mar       Apr       May       Jun       Jul       Aug
1987   1664.81   2397.53   2840.71   3547.29   3752.96   3714.74   4349.61   3566.34
1988   2499.81   5198.24   7225.14   4806.03   5900.88   4951.34   6179.12   4752.15
1989   4717.02   5702.63   9957.58   5304.78   6492.43   6630.80   7349.62   8176.62
1990   5921.10   5814.58  12421.25   6369.77   7609.12   7224.75   8121.22   7979.25
1991   4826.64   6470.23   9638.77   8821.17   8722.37  10209.48  11276.55  12552.22
1992   7615.03   9849.69  14558.40  11587.33   9332.56  13082.09  16732.78  19888.61
1993  10243.24  11266.88  21826.84  17357.33  15997.79  18601.53  26155.15  28586.52
          Sep       Oct       Nov       Dec
1987   5021.82   6423.48   7600.60  19756.21
1988   5496.43   5835.10  12600.08  28541.72
1989   8573.17   9690.50  15151.84  34061.01
1990   8093.06   8476.70  17914.66  30114.41
1991  11637.39  13606.89  21822.11  45060.69
1992  23933.38  25391.35  36024.80  80721.71
1993  30505.41  30821.33  46634.38 104660.67
```

2.3 時系列のプロット

時系列データを読み込んだ後，次に行うべきことはデータを図に描くことである．これは，R の関数の plot.ts() を用いて実行できる．

例えば，イングランドの 42 名の歴代の王の享年の推移グラフを作成するには，次のようにする．

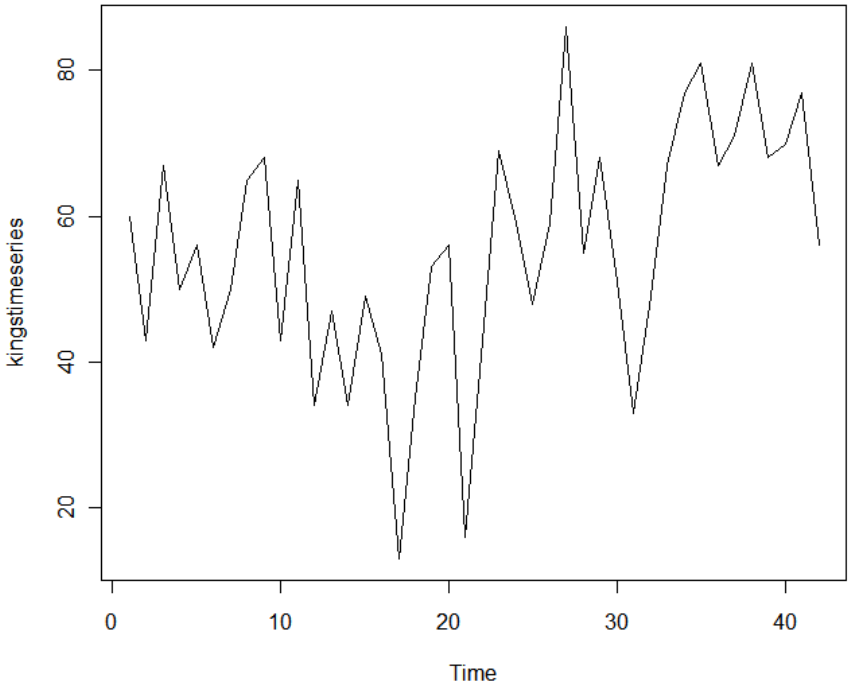
```
> plot.ts(kingstimeseries)
```

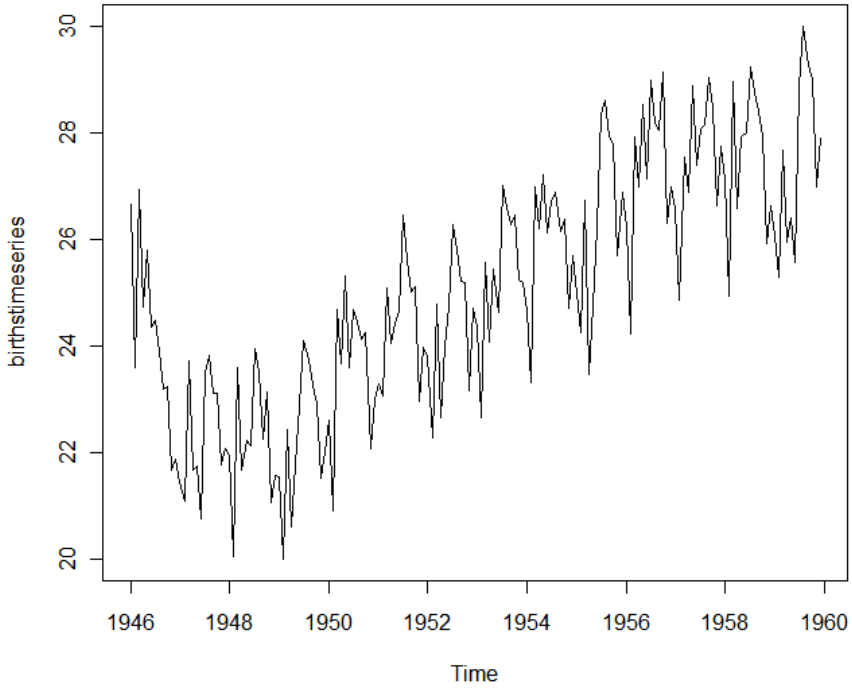
推移グラフより，この時系列は加法モデルを利用して記述できそうであることがわかる．なぜなら，ランダムな変動が時間の推移に対してほぼ一定であるからである．

同様に，ニューヨークの出生数の推移グラフを作成するには次のようにする．

```
> plot.ts(birthstimeseries)
```

この推移グラフより，月次出生数の変動には季節変動がありそうであることがわかる．なぜなら，夏には常に山となり，冬には常に谷となっているからである．また，この時系列データも加法モデルで記述できそうである．なぜなら，季節変動はほぼ一定であり，時系列の水準に依存していそうになく，不規則な変動もほぼ一定のように見えるからである．

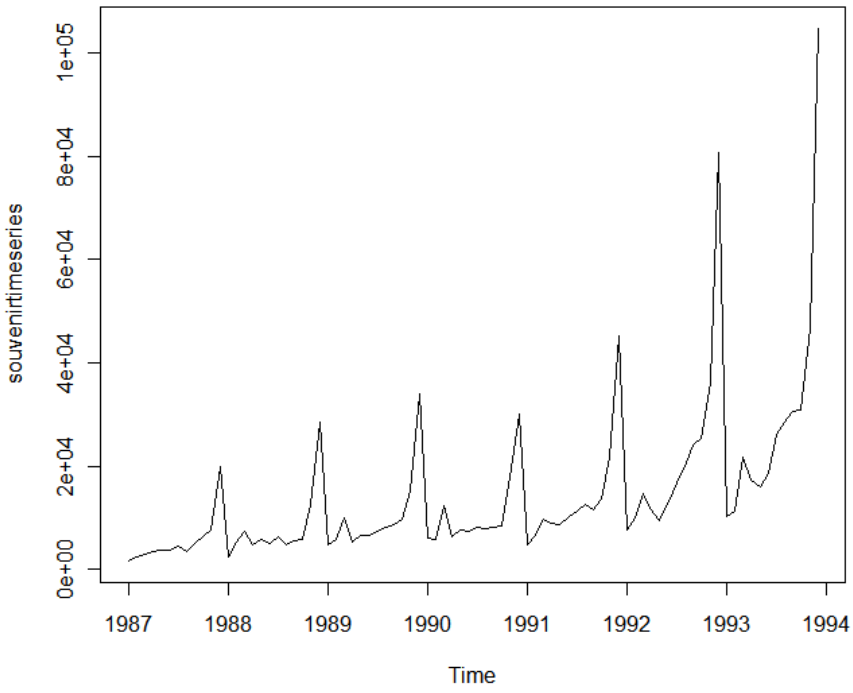




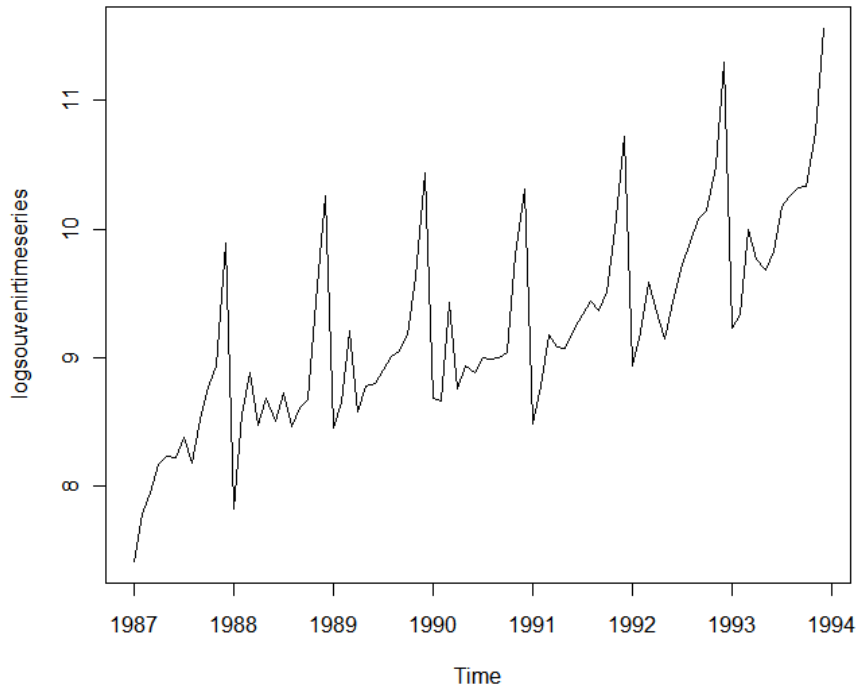
同様に，オーストラリアのクイーンズランドのリゾートタウンにおける土産物店の月次売り上げデータの推移グラフを作成するには，次のようにする．

```
> plot.ts(souvenirtimeseries)
```

この場合，加法モデルは適切ではないことがわかる．なぜなら，季節変動と不規則変動は時系列の水準の変化によって増加しているからである．だから，加法モデルで記述できるようにデータを変換するほうが良いかもしれない．例えば，オリジナルデータの自然対数変換を行い，推移グラフを作成するには次のようにする．




```
> logsouvenirtimeseries <- log(souvenirtimeseries)
> plot.ts(logsouvenirtimeseries)
```



対数変換した時系列の季節変動と不規則変動の大きさは，時間においてほぼ一定であり，時系列のレベルに依存しないことがわかる．だから，この対数変換した時系列は，加法モデルを利用して記述できる．

2.4 時系列の分解

時系列の分解は，構成要素，通常は，トレンド成分，不規則成分への分解を，季節性の時系列の場合は，これらに加えて季節成分という構成要素への分解である．

2.4.1 非季節データの分解

季節性を持たない時系列は，トレンド成分と不規則成分から構成される．時系列の分解は，トレンド成分と不規則成分とを推定することにより時系列を成分に分解することである．

加法モデルで記述できる非季節時系列のトレンド成分を推定するには，時系列の単純移動平均を計算するといった方法が一般的である．

R の “TTR” パッケージにある SMA() 関数を利用して，単純移動平均による時系列データの平滑化を行うことができる．この関数を利用するには，先ず “TTR” パッケージをインストールする必要がある（R のパッケージのインストールの方法については，1.3 節の「R のパッケージのインストール法」を参照のこと）．一度 “TTR” パッケージをインストールしておくと，次を入力することによりこれをロードすることができる．

```
> library("TTR")
```

これにより，時系列データを平滑化するために “SMA()” 関数を利用することができる．SMA() 関数を利用するとき，パラメータ ‘n’ を用いて単純移動平均の次数（スパン）を指定する必要がある．例えば，次数 5 の単純移動平均を求めるには，SMA() 関数で `n=5` を指定する．

例えば，記述のイングランドの歴代 42 名の王の享年の時系列は非季節的であり，データのランダムな変動はほぼ同じ大きさなので，加法モデルにより記述できる．

だから，単純移動平均を用いて時系列のトレンド成分を推定することができる．次数 3 の単純移動平均で平滑化し，平滑化された時系列データをグラフ化するには次を入力する．

```
> kingstimeseriesSMA3 <- SMA(kingstimeseries, n=3)
> plot.ts(kingstimeseriesSMA3)
```

次数 3 の単純移動平均による平滑化では，依然として不規則変動が大きいことがわかる．だから，より大きな次数で平滑化する必要がある．次数の決定には試行錯誤が必要である．例えば，次数 8 の単純移動平均を行ってみよる．

```
> kingstimeseriesSMA8 <- SMA(kingstimeseries, n=8)
> plot.ts(kingstimeseriesSMA8)
```

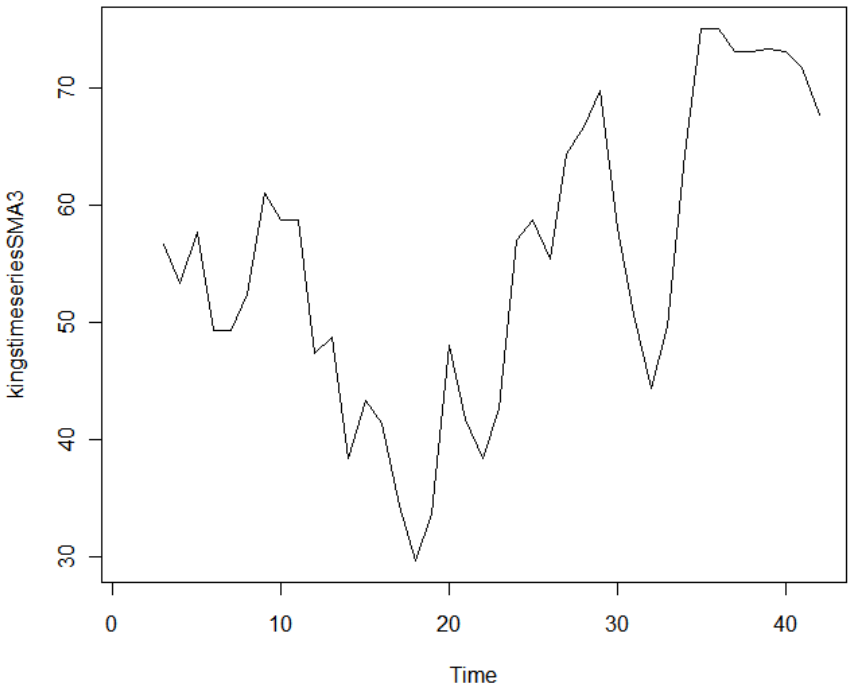
次数 8 の単純移動平均により平滑化したデータは，明確なトレンド成分を持つことを示している．つまり，イングランドの王の享年は，最初の 20 名の間に 55 歳から 38 歳に減少し，その後，40 代の治政までに 73 歳まで増加している．

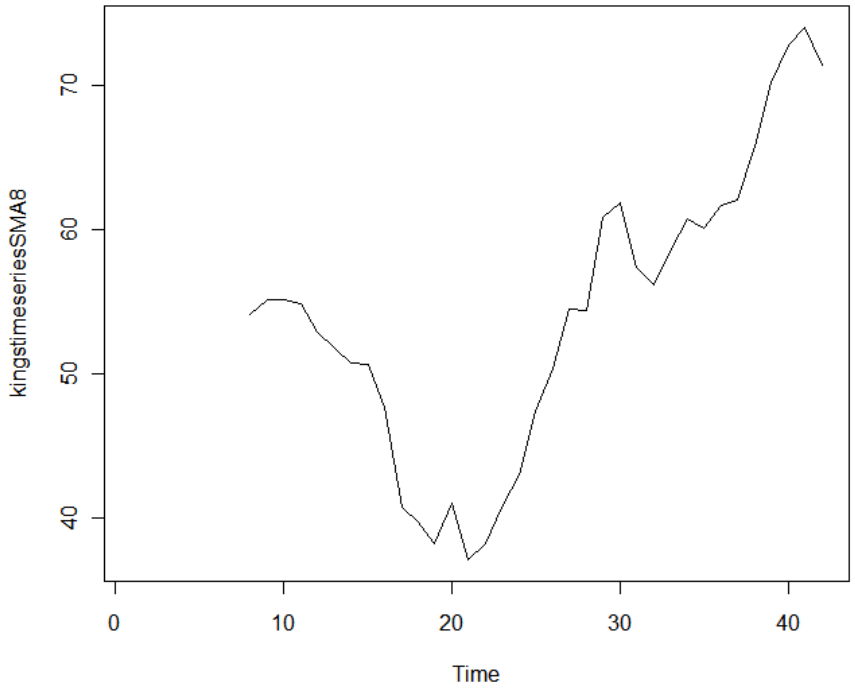
2.4.2 季節的データの分解

季節性を持つ時系列データは，トレンド成分，季節成分，不規則成分から構成される．時系列の分解は，これらの成分への分解を意味する．すなわち，これらの成分を推定することである．

加法モデルにより記述できる季節的時系列のトレンド成分と季節成分を推定するには，R の “decompose()” 関数を利用することができる．この関数は，加法モデルで記述できる時系列のトレンド成分，季節成分，不規則成分を推定する．

関数 “decompose()” は結果をリストオブジェクトとして返す．このオブジェクトには，季節成分，トレンド成分，不規則成分が，それぞれ，“seasonal”，“trend”，“random” という名前の要素として保





存されている．

例えば，既に述べたように，ニューヨークにおける出生数の時系列は夏に多く，冬に少なくなるという季節性を持ち，おそらく，加法モデルにより記述できる．なぜなら，この時系列の季節成分と不規則成分は，時間が経過してもほぼ一定であるからである．この時系列のトレンド成分，季節成分，不規則成分を推定するには，次を入力する．

```
> birthstimeseriescomponents <- decompose(birthstimeseries)
```

季節成分，トレンド成分，不規則成分の推定値は，それぞれ，変数 birthstimeseriescomponents$seasonal, birthstimeseriescomponents$trend，birthstimeseriescomponents$random に保存さ

れている．例えば，季節成分を表示するには次を入力する．

```
> birthstimeseriescomponents$seasonal # 季節成分の推定値の取得
           Jan        Feb        Mar        Apr        May        Jun        Jul        Aug
1946 -0.6771947 -2.0829607  0.8625232 -0.8016787  0.2516514 -0.1532556  1.4560457  1.1645938
1947 -0.6771947 -2.0829607  0.8625232 -0.8016787  0.2516514 -0.1532556  1.4560457  1.1645938
1948 -0.6771947 -2.0829607  0.8625232 -0.8016787  0.2516514 -0.1532556  1.4560457  1.1645938
1949 -0.6771947 -2.0829607  0.8625232 -0.8016787  0.2516514 -0.1532556  1.4560457  1.1645938
1950 -0.6771947 -2.0829607  0.8625232 -0.8016787  0.2516514 -0.1532556  1.4560457  1.1645938
1951 -0.6771947 -2.0829607  0.8625232 -0.8016787  0.2516514 -0.1532556  1.4560457  1.1645938
1952 -0.6771947 -2.0829607  0.8625232 -0.8016787  0.2516514 -0.1532556  1.4560457  1.1645938
1953 -0.6771947 -2.0829607  0.8625232 -0.8016787  0.2516514 -0.1532556  1.4560457  1.1645938
1954 -0.6771947 -2.0829607  0.8625232 -0.8016787  0.2516514 -0.1532556  1.4560457  1.1645938
1955 -0.6771947 -2.0829607  0.8625232 -0.8016787  0.2516514 -0.1532556  1.4560457  1.1645938
1956 -0.6771947 -2.0829607  0.8625232 -0.8016787  0.2516514 -0.1532556  1.4560457  1.1645938
1957 -0.6771947 -2.0829607  0.8625232 -0.8016787  0.2516514 -0.1532556  1.4560457  1.1645938
1958 -0.6771947 -2.0829607  0.8625232 -0.8016787  0.2516514 -0.1532556  1.4560457  1.1645938
1959 -0.6771947 -2.0829607  0.8625232 -0.8016787  0.2516514 -0.1532556  1.4560457  1.1645938
           Sep        Oct        Nov        Dec
1946  0.6916162  0.7752444 -1.1097652 -0.3768197
1947  0.6916162  0.7752444 -1.1097652 -0.3768197
1948  0.6916162  0.7752444 -1.1097652 -0.3768197
1949  0.6916162  0.7752444 -1.1097652 -0.3768197
1950  0.6916162  0.7752444 -1.1097652 -0.3768197
1951  0.6916162  0.7752444 -1.1097652 -0.3768197
1952  0.6916162  0.7752444 -1.1097652 -0.3768197
1953  0.6916162  0.7752444 -1.1097652 -0.3768197
1954  0.6916162  0.7752444 -1.1097652 -0.3768197
1955  0.6916162  0.7752444 -1.1097652 -0.3768197
1956  0.6916162  0.7752444 -1.1097652 -0.3768197
1957  0.6916162  0.7752444 -1.1097652 -0.3768197
1958  0.6916162  0.7752444 -1.1097652 -0.3768197
1959  0.6916162  0.7752444 -1.1097652 -0.3768197
```

季節成分は，1 月から 12 月に対して推定され，各年同じ値である．最大の季節成分は 7 月（約 1.46）であり，最小は 2 月（約 −2.08）であることから，7 月が出生の山，2 月が谷であることを示している．

推定されたトレンド成分，季節成分，不規則成分を，“plot()” 関数を利用してグラフ化することができる．

```
> plot(birthstimeseriescomponents)
```

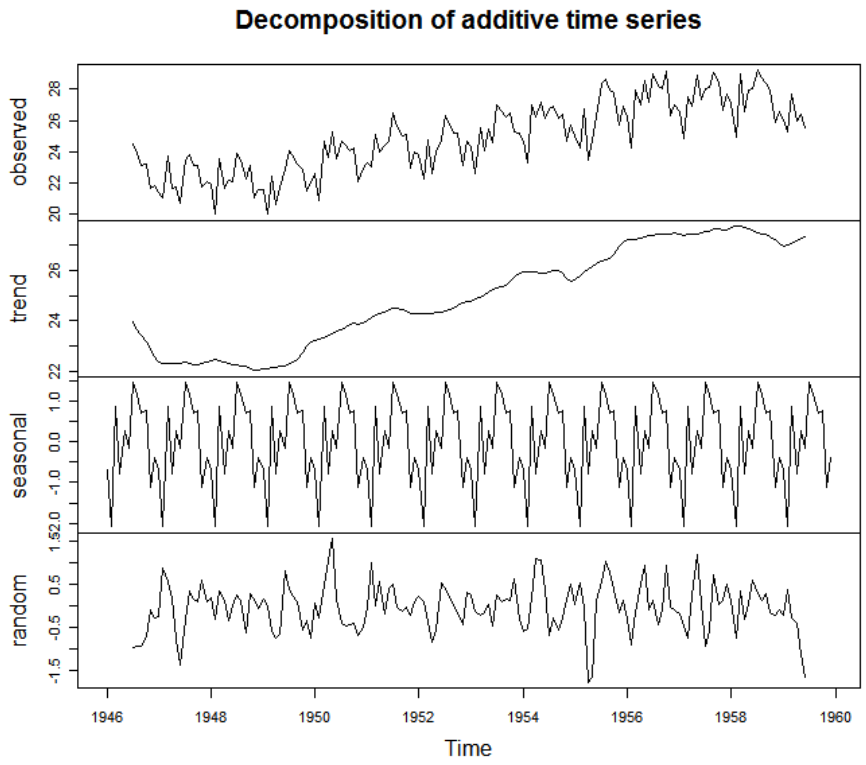
上図の一番上は原系列のグラフであり，上から 2 つ目がトレンド成分の，上から 3 番目が季節成分の，一番下が不規則成分のグラフである．トレンド成分の推定値は，1947 年の約 24 から 1948 年の約 22 まで減少し，その後，1959 年の約 27 まで増加していることがわかる．

2.4.3 季節調整

加法モデルを用いて記述できる季節性の時系列があるとき，原系列の季節成分を推定することにより時系列を調整することができる．これは，“decompose()” 関数を用いて計算した季節成分の推定値を用いて実行可能である．

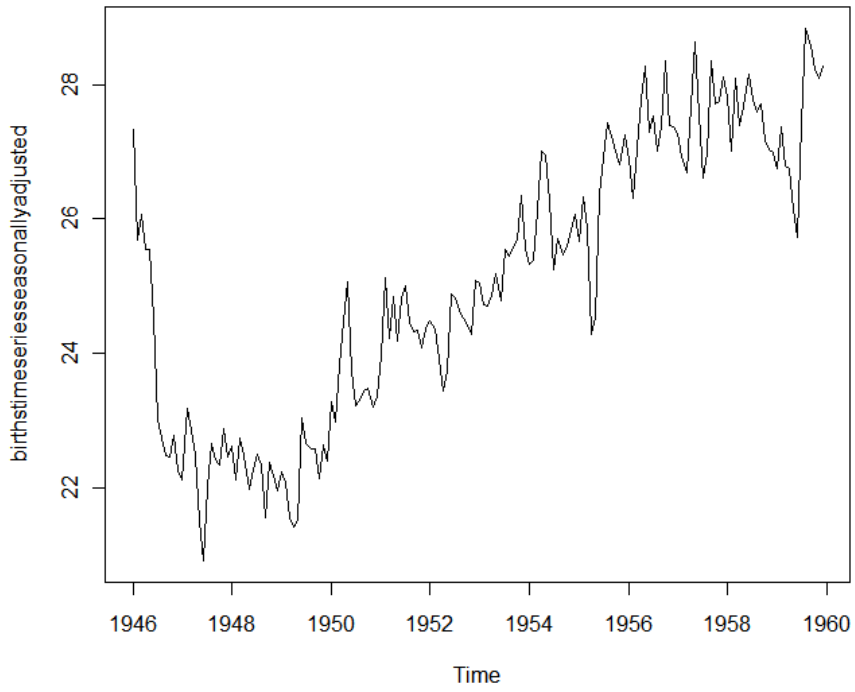
例えば，ニューヨークの出生数の時系列データを季節調整するには，“decompose()” 関数を利用して季節成分を推定し，原系列から季節成分を引く．

```
> birthstimeseriescomponents <- decompose(birthstimeseries)
> birthstimeseriesseasonallyadjusted <- birthstimeseries - birthstimeseriescomponents$seasonal
```



季節調整した結果をグラフ化するには，“plot()” 関数を次の形で利用する．

```
> plot(birthstimeseriesseasonallyadjusted)
```



2.5　指数平滑法による予測

時系列データの短期予測を行うには，指数平滑法を利用すればよい．

2.5.1 単純指数平滑法

水準が一定で，加法モデルで記述できる季節性を持たない時系列の場合，指数平滑法により短期予測を行うことができる．

単純指数平滑法は，現在の時点における水準を推定する方法を与える．当該時点の推定のために平滑化の度合いをパラメータ *alpha* によりコントロールする．*alpha* は，0 以上，1 以下の値を取る．*alpha* が 0 に近いとき，将来の値を予測するときに直近の観測値をあまり重視しないことを意味する．

例えば，ファイル `http://robjhyndman.com/tsdldata/hurst/precip1.dat` は，1813 年から 1912 年までのロンドンにおける年間降雨量データである（オリジナルは，Hipel and McLeod, 1994）．次により，このデータを R に読み込むことができる．

```
> rain <- scan("http://robjhyndman.com/tsdldata/hurst/precip1.dat",skip=1)
Read 100 items
> rainseries <- ts(rain,start=c(1813))
> plot.ts(rainseries)
```

図より，水準がほぼ一定（平均が 25 インチでほぼ一定）であることがわかる．この時系列のランダムな変動もほぼ一定なので，加法モデルを利用してこのデータを記述できる．だから，単純指数平滑法を用いて予測する．

R で単純指数平滑法による予測を行うには，R の “HoltWinters()” 関数を用いて指数平滑予測モデルに当てはめる．HoltWinters() 関数により単純指数平滑化を行うには，関数のパラメータとして `beta=FALSE` および `gamma=FALSE` を指定する（パラメータ *beta*，*gannma* は，後で説明するように，Holt（ホルト）の指数平滑法または Holt-Winters（ホルト・ウィンタース）法で用いる）．

HoltWinters() 関数は，いくつかの名前付き要素を持つリスト変数を返す．

例えば，ロンドンにおける年間降雨量時系列データの予測を単純指数平滑法で行うには，次のようにする．

```
> rainseriesforecasts <- HoltWinters(rainseries, beta=FALSE, gamma=FALSE)
> rainseriesforecasts
Smoothing parameters:
alpha: 0.02412151
beta : FALSE
gamma: FALSE
Coefficients:
[,1]
a 24.67819
```

HoltWinters() の出力より，*alpha* パラメータの推定値は約 0.024 であることがわかる．これは 0 に非常に近いので，予測は直近のおよびそれほど最近ではない観測値の両者を重視していることがわかる（どちらかというと，直近の観測値の方の重みが大きいが）．

HoltWinters() 関数は，デフォルトでは原系列がカバーする期間での予測のみを行う．ロンドンの降雨量時系列の場合，1813 年から 1912 年までのデータなので，予測も 1813 年から 1912 年までである．

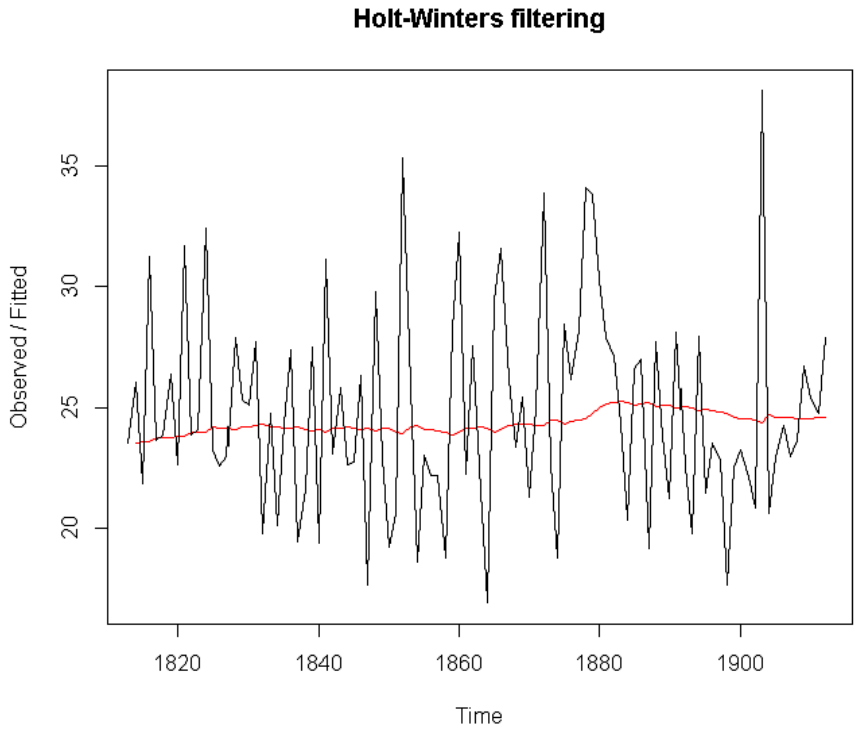
上記の例では，HoltWinters() の出力をリスト変数 “rainseriesforecasts” に保存している．HoltWinters() による予測はこのリスト変数の “fitted” という名前の要素に保存されているので，次により表示することができる．

```
> rainseriesforecasts$fitted
Time Series:
Start = 1814
End = 1912
Frequency = 1
```

```
xhat level
1814 23.56000 23.56000
1815 23.62054 23.62054
1816 23.57808 23.57808
1817 23.76290 23.76290
1818 23.76017 23.76017
1819 23.76306 23.76306
1820 23.82691 23.82691
...
1905 24.62852 24.62852
1906 24.58852 24.58852
1907 24.58059 24.58059
1908 24.54271 24.54271
1909 24.52166 24.52166
1910 24.57541 24.57541
1911 24.59433 24.59433
1912 24.59905 24.59905
```

原系列と一緒に予測をグラフ化するには次を入力する．

```
> plot(rainseriesforecasts)
```



図では，原系列は黒色の線で，予測は赤色の線で表示されている，予測の時系列は，原系列と比べてはるかになめらかであることがわかる

予測の正確さの尺度として，標本内（$in-sample$）予測誤差の 2 乗和，つまり，原系列の期間における予測誤差を利用することができる．

```
> rainseriesforecasts$SSE
[1] 1828.855
```

よって，平方誤差の和は 1828.855 である．

単純指数平滑法では時系列の最初のデータを水準の初期値として利用することが多い．例えば，ロン

ドンの降雨量データでは，最初のデータは 1813 年の 23.56（インチ）である．HoltWinters() 関数では，‘verb—l.start—’ パラメータを利用して水準の初期値を指定することができる．例えば，水準の初期値を 23.56 に指定して予測するには，次のようにする．

```
> HoltWinters(rainseries, beta=FALSE, gamma=FALSE, l.start=23.56)
Holt-Winters exponential smoothing without trend and without seasonal component.

Call:
 HoltWinters(x = rainseries, beta = FALSE, gamma = FALSE, l.start = 23.56)

Smoothing parameters:
 alpha:  0.02412151
 beta :  FALSE
 gamma:  FALSE

Coefficients:
      [,1]
a 24.67819
```

既に述べたように，HoltWinters() 関数のデフォルトでは，原系列がカバーする期間の予測のみを行う．降雨量の時系列では，この期間は 1813 年から 1912 年までである．予測期間を拡大するには，R の “forecast” パッケージにある “forecast.HoltWinters()” を利用することができる．これを利用するには，まず “forecast” をインストールしておく必要がある（パッケージのインストールについては，1.3 節の「R のパッケージのインストール法」を参照のこと）．

“forecast” パッケージをインストール後，次により “forecast” パッケージをロードする．

```
> library("forecast")
```

forecast.HoltWinters() 関数を用いるとき，その最初の引数（input）として，HoltWinters() モデルで作成した予測モデルを与える．降雨量の時系列では，HoltWinters() で作った予測モデルを変数 “rainseriesforecasts” に保存している．forecast.HoltWinters() 関数で ‘h’ パラメータを指定することにより，さらに予測したい時点数を指定することができる．例えば，降雨量の時系列で 1814 年から 1820 年までの予測を行うには（8 年分多く），次を入力する．

```
> rainseriesforecasts2 <- forecast.HoltWinters(rainseriesforecasts, h=8)
> rainseriesforecasts2
     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
1913       24.67819 19.17493 30.18145 16.26169 33.09470
1914       24.67819 19.17333 30.18305 16.25924 33.09715
1915       24.67819 19.17173 30.18465 16.25679 33.09960
1916       24.67819 19.17013 30.18625 16.25434 33.10204
1917       24.67819 19.16853 30.18785 16.25190 33.10449
1918       24.67819 19.16694 30.18945 16.24945 33.10694
1919       24.67819 19.16534 30.19105 16.24701 33.10938
1920       24.67819 19.16374 30.19265 16.24456 33.11182
```

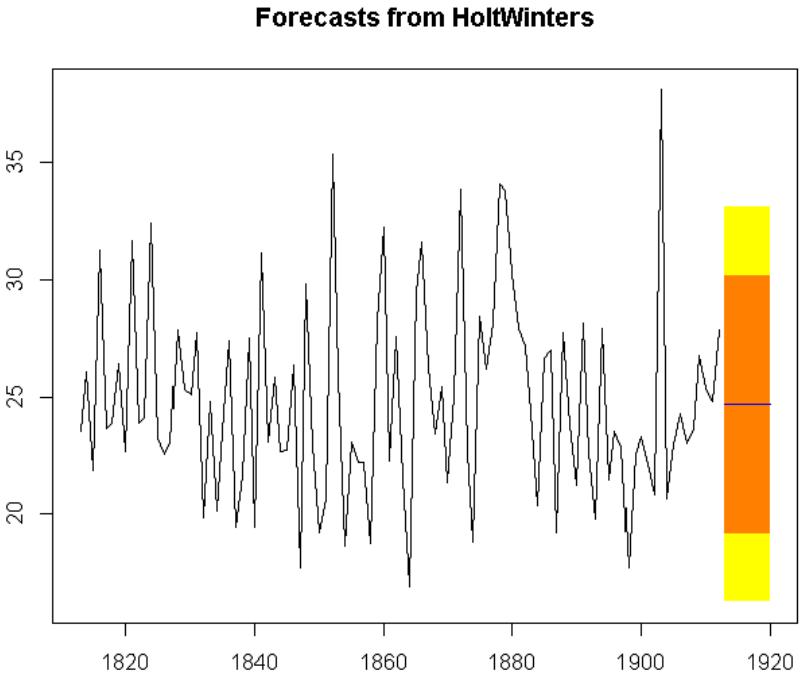
forecast.HoltWinters() 関数は，各時点の予測に対して 80% および 95% 予測区間を与える．例えば，1920 年の予測降雨量は 24.68 インチであり，95% 予測区間は $(16.24, 33.11)$ である．

forecast.HoltWinters() 関数による予測をグラフ化するには，“plot.forecast()” 関数を用いる．

```
> plot.forecast(rainseriesforecasts2)
```

図では，1913 年から 1920 年の予測は青色の線で，80% 予測区間はオレンジ色の領域，95% 予測区間は黄色の斜域で表示されている．

‘予測誤差’ は，各観測時点に対して “観測値 − 予測値” で計算される．予測誤差を計算できるのは，



原系列がカバーする期間（降雨量データでは 1813 年から 1912 年まで）のみである．既に述べたように，予測モデルの精度の尺度の 1 つとして，標本内予測誤差に対する誤差の 2 乗和（SSE）がある．

標本内予測誤差は，forecast.HoltWinters() 関数によって返されるリスト変数の “residual” という名前の要素に保存されている．予測モデルをそれ以上改良できないということは，予測値と予測誤差の間に相関がないということであるべきである．言い換えると，連続した予測値と予測誤差の間に相関がある場合，単純指数平滑法による予測は，他の予測手法により改良することができる可能性がある．

これが正しいかどうかを確認するために，標本内予測誤差をラグ $1-20$ に対するコレログラムを用いることができる．“acf()” 関数を用いて予測誤差のコレログラムを計算することができる．調べたいラグの最大を指定するには，acf() において ‘lag.max’ パラメータを利用する．

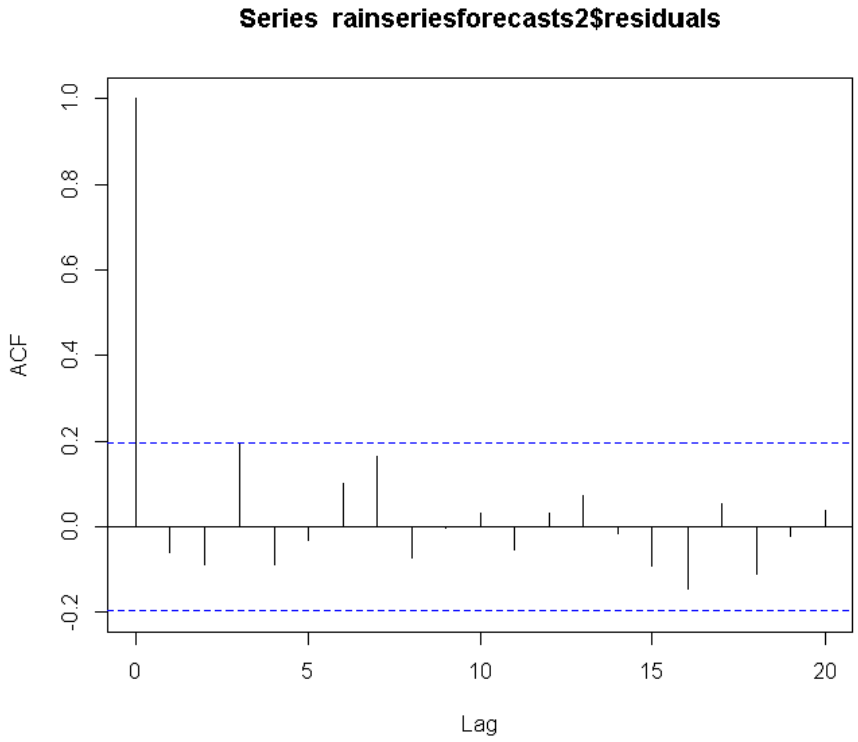
例えば，ロンドンの降雨量データに対して，ラグ $1-20$ をとして標本内予測誤差のコレログラムを計算するには，次のようにする．

```
> acf(rainseriesforecasts2$residuals, lag.max=20)
```

標本コレログラムより，ラグ 3 での自己相関が有意水準の限界に接触していることがわかる．ラグ $1-20$ に対する無相関の検定が有意かどうかを検証するために， Ljung-Box 検定を行うことができる．これは，“Box.test()” 関数を用いて実行できる．調べたい最大ラグは，Box.test() 関数の ‘lag’ パラメータで指定する．例えば，ロンドンの降雨量データの標本内予測誤差に対して，ラグ $1-20$ に対して 0 ではない自己相関があるかどうかを検定するには，次のようにする．

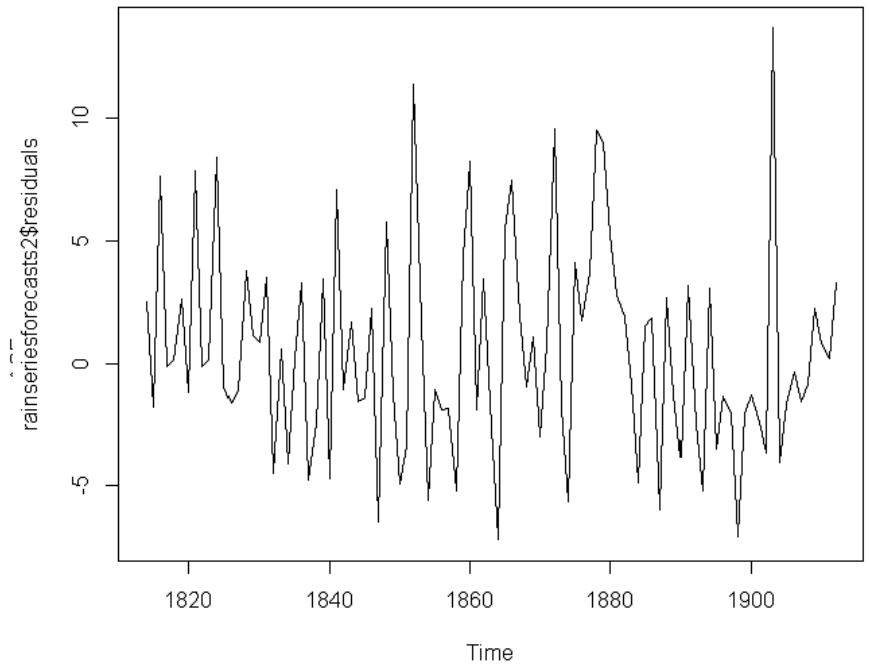
```
> Box.test(rainseriesforecasts2$residuals, lag=20, type="Ljung-Box")

        Box-Ljung test

data:  rainseriesforecasts2$residuals
X-squared = 17.4008, df = 20, p-value = 0.6268
```

Ljung-Box 検定統計量の値は 17.4 であり，p 値は 0.6 である．よって，ラグ $1-20$ に対する標本内予測誤差において自己相関があるという根拠はほぼない．



予測モデルをより改良できるかどうかを確認するために，予測誤差が平均 0，分散が一定の正規分布に従っているかどうかを確認することはよいアイデアである．予測誤差の分散が一定かどうかを確認するには，標本内予測誤差の推移グラフを作成する．

```
> plot.ts(rainseriesforecasts2$residuals)
```



図より，標本内予測誤差の分散はほぼ一定であるが，時系列のスタート時点（1820 年から 1830 年）における変動の大きさが，後の時点と比べて少し小さいことがわかる．

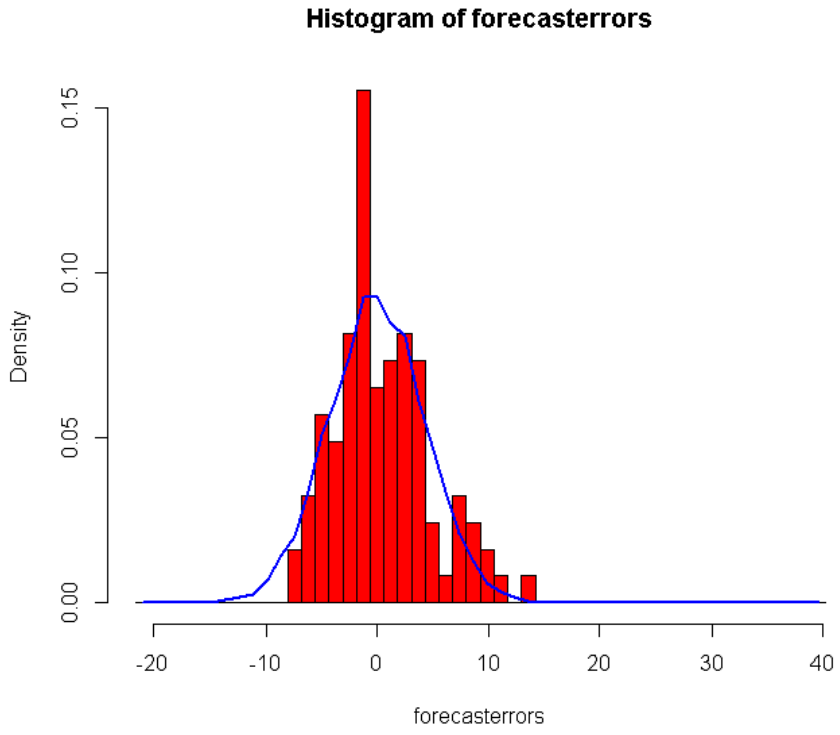
予測誤差のヒストグラムに，平均が 0 で，標本分散と同じ分散を持つ正規分布を重ねて描くことによ

り，予測誤差が平均が 0 の正規分布に従っているかどうかを確認することができる．このために，次に示す “plotForecastErrors” 関数を定義する．

```
> plotForecastErrors <- function(forecasterrors)
  {
     # 予測誤差のヒストグラム（赤色）の作成
     mybinsize <- IQR(forecasterrors)/4
     mymin  <- min(forecasterrors)*3
     mymax  <- max(forecasterrors)*3
     mybins <- seq(mymin, mymax, mybinsize)
     hist(forecasterrors, col="red", freq=FALSE, breaks=mybins)
     # ヒストグラムの下の面積を 1 とするために，freq=FALSE とする
     mysd   <- sd(forecasterrors)
     # 平均が 0，標準偏差が mysd の正規分布に従うデータを生成
     mynorm <- rnorm(10000, mean=0, sd=mysd)
     myhist <- hist(mynorm, plot=FALSE, breaks=mybins)
     # 予測誤差のヒストグラムに正規分布のカーブを重ねて描く
     points(myhist$mids, myhist$density, type="l", col="blue", lwd=2)
  }
```

この関数を利用するには，R にコピー&ペーストする必要がある．その後，plotForecastErrors() を用いて，降雨量の予測に対する予測誤差のヒストグラムを（正規分布を重ねて）描くことができる．

```
> plotForecastErrors(rainseriesforecasts2$residuals)
```



図より，予測誤差の分布の中心はほぼ 0 であるが，正規分布と比べると少し右に歪んで分布している．しかし，歪みはかなり小さく，予測誤差は平均が 0 の正規分布であると判断できる．

Ljung-Box 検定から，標本内自己相関は 0 ではないという証拠はほとんど見られず，予測誤差の分布は平均が 0 の正規分布と判断してよいことがわかった．このことより，単純指数平滑法はロンドンの降雨量に対する適切な予測モデルを与えており，これ以上改良できないことを示唆する．さらに，80%，90% 予測区間に関する仮定（予測誤差には自己相関がなく，予測誤差は平均が 0，分散が一定の正規分

布に従うというもの）もおそらく有効である．

2.5.2 Holt の指数平滑法

増加または減少のトレンドを持ち，加法モデルを用いて記述できる季節性を持たない時系列があるとき，Holt の指数平滑法を用いて短期予測を行うことができる．

Holt の指数平滑法は，現在の時点における水準とスロープを推定する．平滑化は 2 つのパラメータにより調整できる．1 つは `alpha` であり，これは観測時点での水準を推定するためのものであり，もう 1 つの `beta` は，観測時点におけるトレンド成分の傾き b を推定するためのものである．

トレンドを持ち，加法モデルで記述できる季節性を持たない時系列の例として，1866 年から 1911 年の女性のスカートの裾の直径の年次データがある．このデータは，ファイル `http://robjhyndman.com/tsdldata/roberts/skirts.dat` より利用可能である（原データは，Hipel and McLeod, 1994 による）．

次により，このデータを読み込み，グラフ化することができる．

```
> skirts <- scan("http://robjhyndman.com/tsdldata/roberts/skirts.dat",skip=5)
> skirtsseries <- ts(skirts,start=c(1866))
> plot.ts(skirtsseries)
```

図より，1866 年から 1880 年にかけて直径は 600 から 1050 に増加しており，その後，1911 年の 520 へと減少していることがわかる．

予測するために，HoltWinters() 関数を用いて予測モデルに当てはめることができる．Holt の指数平滑法のために HoltWinters() 関数を利用するには，パラメータ `gamma=FALSE` を指定する（`gamma` パラメータは，Holt-Winter 指数平滑法で用いる）．

例えば，スカートの裾の直径データに対して予測モデルを当てはめるために Holt の指数平滑法を利用するには，次のようにする．

```
> skirtsseriesforecasts <- HoltWinters(skirtsseries, gamma=FALSE)
> skirtsseriesforecasts
Holt-Winters exponential smoothing with trend and without seasonal component.
```


```
Call:
 HoltWinters(x = skirtsseries, gamma = FALSE)

Smoothing parameters:
 alpha:  0.8383481
 beta :  1
 gamma:  FALSE

Coefficients:
        [,1]
a 529.308585
b   5.690464
```

alpha の推定値は 0.84 であり，beta の推定値は 1.00 である．どちらの値も大きい．これは，水準の現在の推定値とトレンド成分の傾き b の推定値の両者が，時系列の直近のデータに依存していることを示す．これは，直感とも一致している．なぜなら，時系列の水準と傾きがしばしば変化しているからである．標本内予測誤差の平方誤差の和は 16954 である．

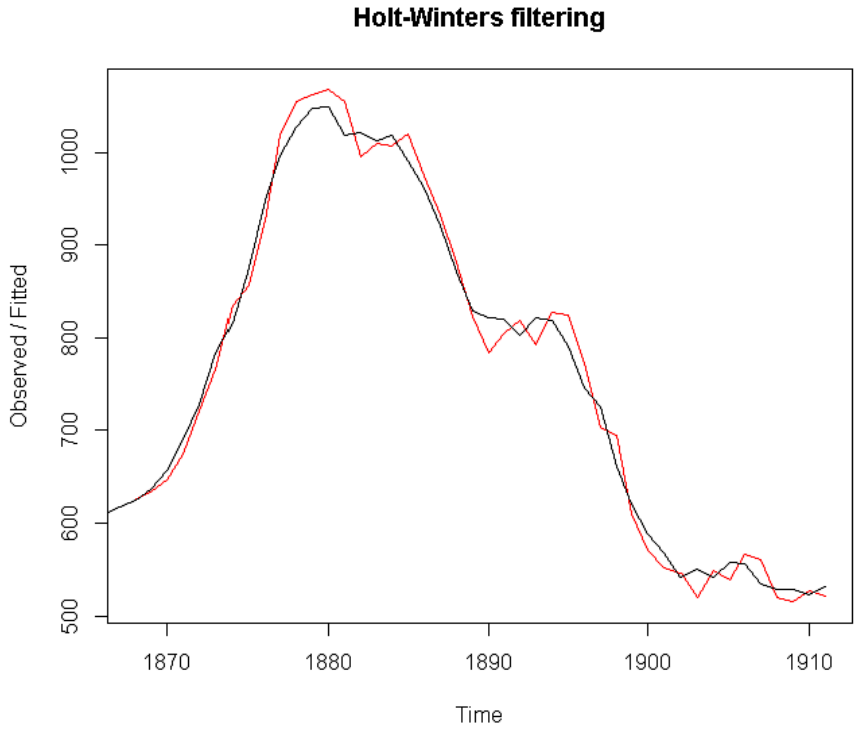
原系列を黒色の線で，予測値を赤色の線でプロットするには，次を入力する．

```
> plot(skirtsseriesforecasts)
```

図より，標本内予測は観測値と非常に良く一致しているが，観測値から少し遅れてずれている．

HoltWinters() 関数で，“l.start” と “b.start” を用いることにより，水準の初期値とトレンド成分の傾き b の初期値を指定することができる．時系列の最初の値（今の場合，スカートデータの 608）を初期値とすることが多い．例えば，Holt の指数平滑法を用いてスカートの裾のデータに予測モデルを当てはめる際，水準の初期値として 608 を，トレンド成分の傾き b の初期値を 9 とするには，次を入力する．

```
> HoltWinters(skirtsseries, gamma=FALSE, l.start=608, b.start=9)
Holt-Winters exponential smoothing with trend and without seasonal component.

Call:
 HoltWinters(x = skirtsseries, gamma = FALSE, l.start = 608, b.start = 9)
```




```
Smoothing parameters:
 alpha:  0.8346775
 beta :  1
 gamma:  FALSE

Coefficients:
        [,1]
a 529.278637
b   5.670129
```

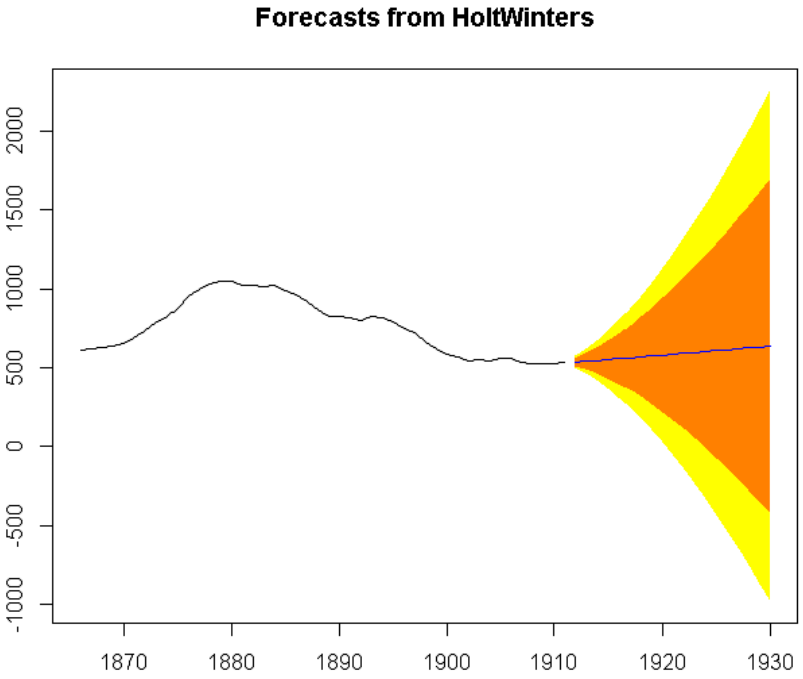
単純指数平滑法に関しては，“forecast” パッケージの forecast.HoltWinters() 関数を用いて原系列がカバーしていない将来の予測を行うことができる．例えば，スカートの裾データは 1866 年から 1911 年までであるが，1912 年から 1930 年まで（追加のデータ点を 19）の予測を行い，プロットするには次のようにする．

```
> skirtsseriesforecasts2 <- forecast.HoltWinters(skirtsseriesforecasts, h=19)
> plot.forecast(skirtsseriesforecasts2)
```

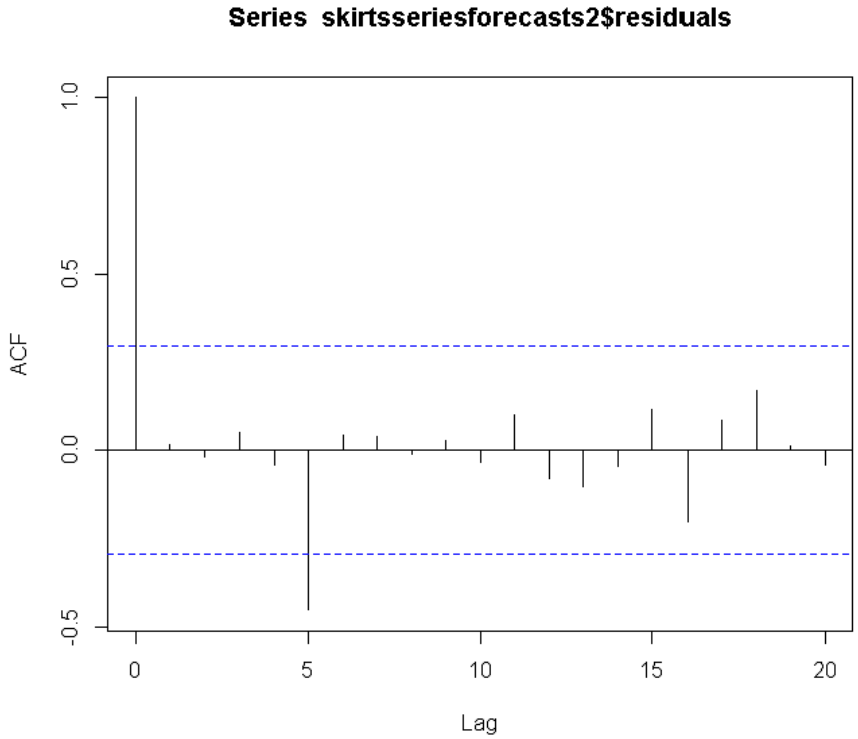
予測値は青色の線で，80% 予測区間はオレンジ色の領域で，95% 予測区間は黄色の領域で示されている．

単純指数平滑法に関して，標本内予測誤差がラグ $1-20$ で 0 でない自己相関を示すかどうかで予測モデルをさらに改良することができるかどうかを確認することができる．例えば，スカートの裾データに対して，コレログラムを作成し，Ljung-Box 検定を行うには，次を入力する．

```
> acf(skirtsseriesforecasts2$residuals, lag.max=20)
> Box.test(skirtsseriesforecasts2$residuals, lag=20, type="Ljung-Box")

        Box-Ljung test

data:  skirtsseriesforecasts2$residuals
```



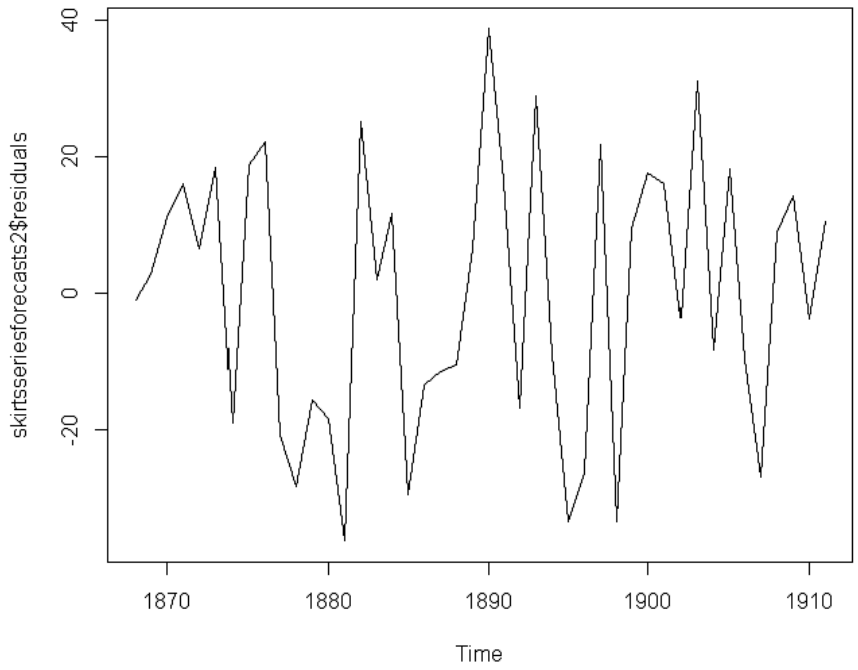

```
X-squared = 19.7312, df = 20, p-value = 0.4749
```

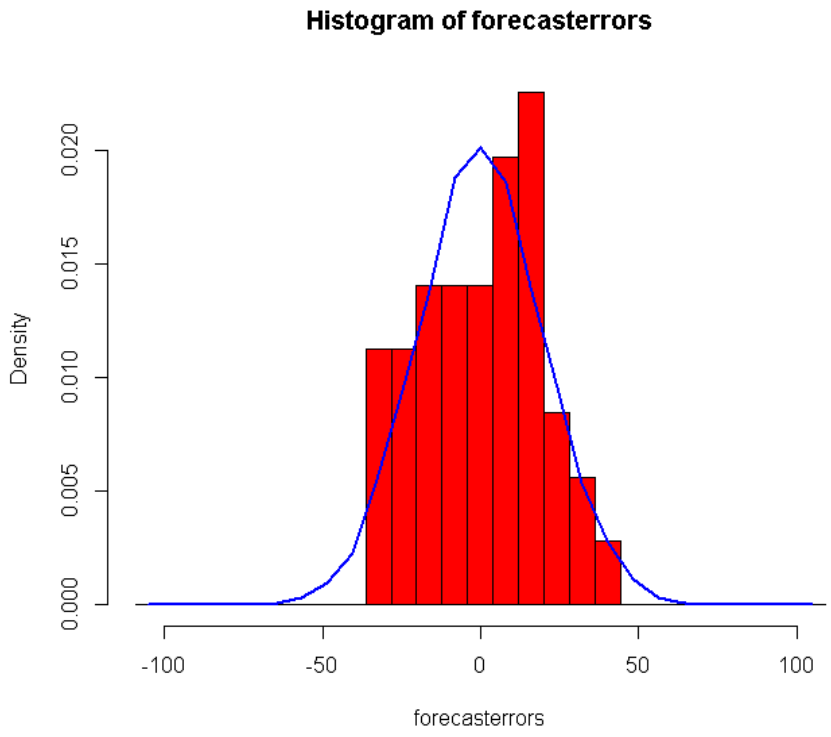


コレログラムより，ラグ 5 での標本内予測誤差に対する標本自己相関が有意性の限界を超えていることがわかる．しかし，ラグを 20 個考えているので，偶然のみでも自己相関が 95% 信頼限界を超える可能性はある．実際，Ljung-Box 検定における p 値は 0.47 であり，これは，ラグ $1-20$ における標本内予測誤差に自己相関があることの証拠とはほとんどならないことを示す．

単純指数平滑法を利用したとき，予測誤差が時間に関して一定であり，平均が 0 の正規分布に従うことを確認すべきである．これには，予測誤差の推移グラフを作成し，予測誤差のヒストグラムに正規分布の密度関数を重ねあわせた図を作成すればよい．

```
> plot.ts(skirtsseriesforecasts2$residuals) # 水グラフ
> plotForecastErrors(skirtsseriesforecasts2$residuals) # ヒストグラムの作成
```





予測誤差の推移グラフより，予測誤差の分散はほぼ一定であることがわかる．予測誤差のヒストグラムより，予測誤差は平均が 0，分散が一定の正規分布であるといってよさそうである．

だから，Ljung-Box 検定は，予測誤差が自己相関を持つとはいえないことを示しており，予測誤差の推移図とヒストグラムは予測誤差の分布は平均 0 の正規分布で，分散は一定であると考えてよいことを示している．よって，Holt の指数平滑法はスカートの裾の直径に対する予測モデルが適切であり，これ以上改良することはできないと結論づけることができる．さらに，80% 予測区間と 95% 予測区間の元にある仮定もおそらく成立していることを意味する．

2.5.3 Holt-Winters の指数平滑法

増加または減少のトレンドと季節性を持つ加法モデルで記述できる時系列に対して，Holt-Winters の指数平滑法により短期予測を行うことができる．

HoltWinters の指数平滑法は，観測時点における水準，傾き，季節の各成分を推定する．平滑度は，*alpha*，*beta*，*gamma* という 3 つのパラメータでコントロールできる．*alpha* は水準の，*beta* はトレンド成分の傾き b の，*gamma* は季節成分の，各期における推定値である．パラメータ *alpha*，*beta*，*gamma* は 0 から 1 の値を取り，0 に近いと将来の値を予測するときに直近の観測値にあまり重みを置かないことを意味する．

トレンドと季節性を持つ加法モデルでおそらく記述できるであろう時系列の例として，オーストラリアのクイーンズランドのビーチ・リゾートにおける土産物屋の月次売上高データを取り上げることができる．

HoltWinters() 関数を用いて，時系列データを予測モデルに当てはめることができる．例えば，土産物屋の月次売上高の対数値に予測モデルを当てはめるには，次のようにする．

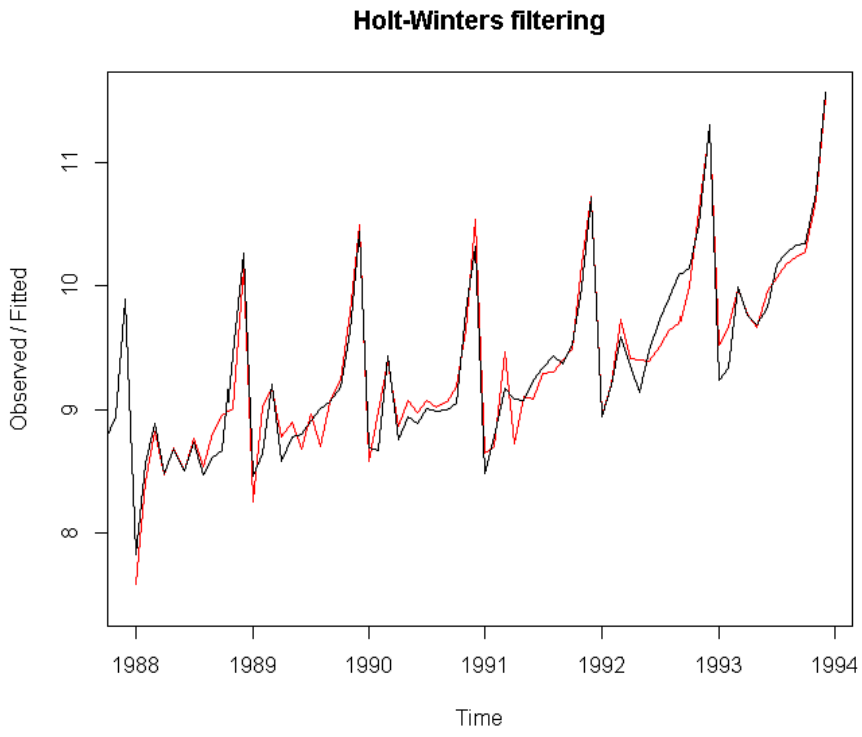
```
> logsouvenirtimeseries <- log(souvenirtimeseries)
> souvenirtimeseriesforecasts <- HoltWinters(logsouvenirtimeseries)
> souvenirtimeseriesforecasts
Holt-Winters exponential smoothing with trend and additive seasonal component.

Call:
 HoltWinters(x = logsouvenirtimeseries)

Smoothing parameters:
 alpha:  0.413418
 beta :  0
 gamma:  0.9561275

Coefficients:
           [,1]
a   10.37661961
b    0.02996319
s1  -0.80952063
s2  -0.60576477
s3   0.01103238
s4  -0.24160551
s5  -0.35933517
s6  -0.18076683
s7   0.07788605
s8   0.10147055
s9   0.09649353
s10  0.05197826
s11  0.41793637
s12  1.18088423
```

```
> souvenirtimeseriesforecasts$SSE
[1] 2.011491
```

alpha, *beta*, *gamma* の推定値は，それぞれ 0.41, 0.00, 0.96 である．*alpha* の推定値（0.41）は，比較的小さく，当期における水準の推定値は，直近の観測値ともっと離れた過去の観測値の両者に依存することを意味する．*beta* の推定値は 0.00 であり，トレンド成分の傾き b の推定値が水準の変化と関係なく初期値とほとんど同じであることを意味する．これは直感的に納得がいく．なぜなら，時系列を通じて水準の変化はほとんど無く，トレンド成分の傾き b はほぼ同じだからである．これに対して，*gamma* の値（0.96）は大きく，当期における奇跡成分の推定値は直近の観測値に大きく依存していることを意味する．

単純指数平滑法と Holt の指数平滑法について，原系列を黒色の線，予測値を赤色の線で 1 つのグラフに表示するには次を入力する．

```
> plot(souvenirtimeseriesforecasts)
```



図より，HoltWinters 指数平滑法は，ほぼ毎年 11 月に生じている季節性の山を非常にうまく予測していることがわかる．

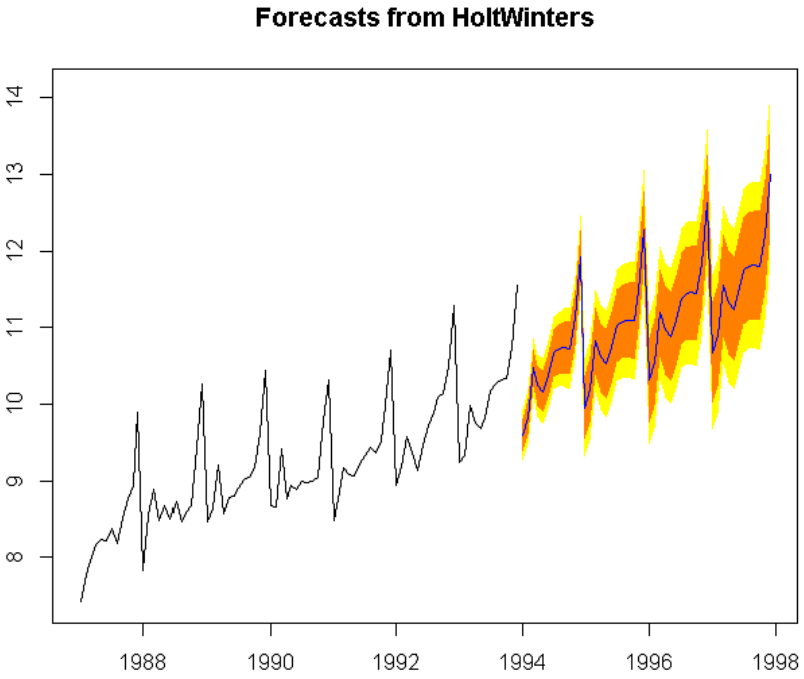
原系列に含まれない将来の期の予測には，“forecast” パッケージにある “forecast.HoltWinters()” 関数を用いる．例えば，土産物店の売り上げの原系列の期間は 1987 年 1 月から 1993 年の 12 月までである．1994 年 1 月から 1998 年 12 月まで（さらに 48 ヶ月）の予測を行い，予測をグラフ化したい場合，次を入力する．

```
> souvenirtimeseriesforecasts2 <- forecast.HoltWinters(souvenirtimeseriesforecasts, h=48)
> plot.forecast(souvenirtimeseriesforecasts2)
```

図で，予測値は青色の線で，オレンジ色と黄色の領域は 80% および 95% 予測区間を示す．

予測モデルに改良の可能性があるかどうかは，コレログラムの作成と，Ljung-Box 検定を用いる標本内予測誤差がラグ $1-20$ で自己相関を持つかどうかを調べることによって確認することができる．

```
> acf(souvenirtimeseriesforecasts2$residuals, lag.max=20)
```




```
> Box.test(souvenirtimeseriesforecasts2$residuals, lag=20, type="Ljung-Box")

        Box-Ljung test

data:  souvenirtimeseriesforecasts2$residuals
X-squared = 17.5304, df = 20, p-value = 0.6183
```

コレログラムより，標本内予測誤差の自己相関はラグ $1-20$ に対する有意性の限界を超えていない．さらに，Ljung-Box 検定の p 値は 0.6 であり，これはラグ $1-20$ における 0 でない自己相関をほとんど示唆しない．

予測誤差は時間の変化に関わらず一定で，平均が 0 の正規分布に従うかどうかを予測誤差の推移図と（正規分布をオーバーレイした）ヒストグラムを作成することにより確認することができる．

```
> plot.ts(souvenirtimeseriesforecasts2$residuals) # 推移グラフの作成
> plotForecastErrors(souvenirtimeseriesforecasts2$residuals) # ヒストグラムの作成
```

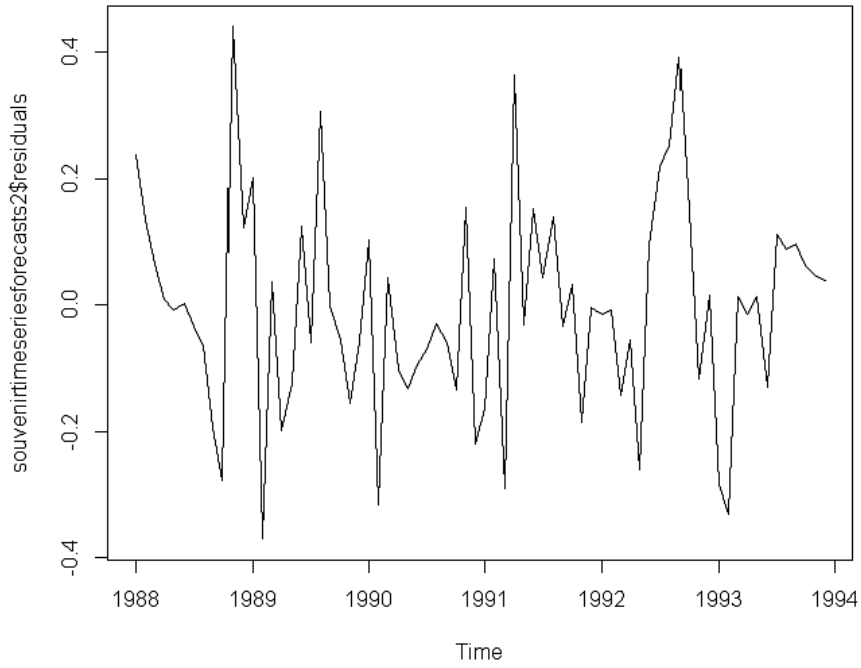
推移グラフより，予測誤差は一定のようである．また，予測誤差のヒストグラムより，予測誤差の分布は平均が 0 の正規分布のようである．

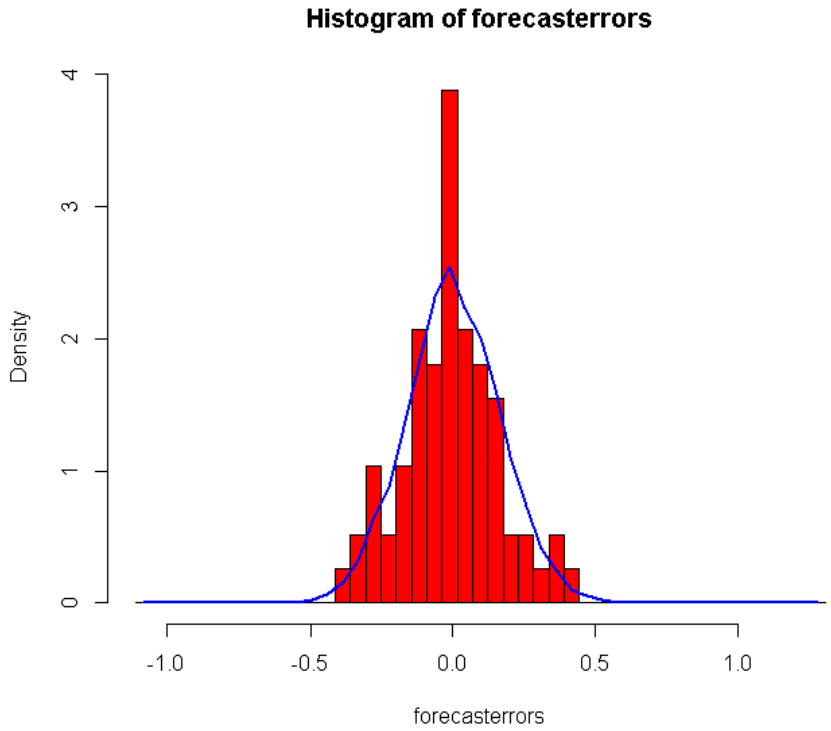
だから，ラグ $1-20$ において自己相関があるという証拠はほとんどなく，予測誤差の分布は平均が 0，分散が一定の正規分布であると考えることができる．このことは，HoltWinters の指数平滑法が土産物屋の売上高の対数の適切な予測モデルを与えており，改良の余地はないことを示唆する．さらに，予測区間の構成の前提条件もおそらく成立しているだろうこともわかる．

2.6 ARIMA モデル

指数平滑法は，予測を行うのに有益で，時系列の連続した値の相関に関する仮定は必要ない．しかし，指数平滑法を用いて行われた予測に対して予測区間を構成したいとき，予測誤差に相関はなく，平均が 0，分散が一定の正規分布に従うという仮定が必要となる．

指数平滑法は，時系列の連続する値の間の相関に関する仮定を必要としないが，説明したいデー





タの相関を考えることにより予測を改良することができる場合がある．自己回帰和分移動平均（Autoregressive Integrated Moving Average : ARIMA）モデルは，時系列の不規則成分に対して明示的に統計モデルを含むので，不規則成分の自己相関が 0 でなくてもよい．

2.6.1 時系列の階差

ARIMA モデルは，定常時系列に対して定義される．よって，非定常時系列に対しては，まず定常化できるまで “階差” を取る必要がある．時系列の階差を d 回取って定常化できるとき，ARIMA(p,d,q) モデルとなる．d は階差を取る回数である．

“diff()” 関数を用いて時系列の階差を取ることができる．例えば，1866 年から 1911 年の女性のスカートの裾の直径の時系列の平均は，年によってかなり変動するので定常ではない．

階差を 1 回取り（“skirtsseries” に保存），階差をプロットするには，次を入力する．

```
> skirtsseriesdiff1 <- diff(skirtsseries, differences=1)
> plot.ts(skirtsseriesdiff1)
```

1 回の階差の平均は定常ではない．よって，さらに階差を取り（2 回目），それが定常性を示すかどうかを確認する．

```
> skirtsseriesdiff2 <- diff(skirtsseries, differences=2)
> plot.ts(skirtsseriesdiff2)
```

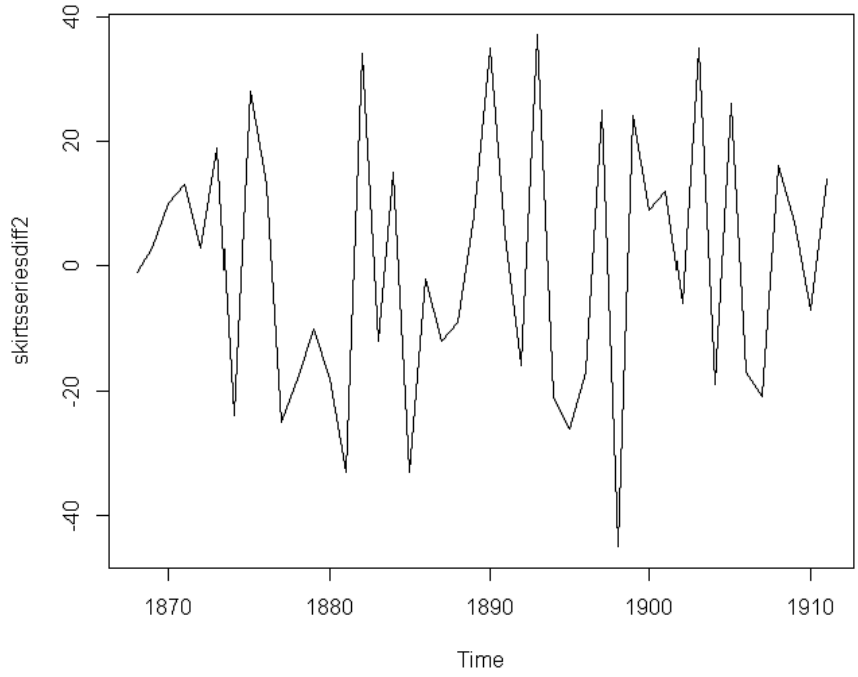
2 回階差を取った時系列は平均と分散において定常性を示している．なぜなら，時系列の水準はほぼ一定であり，系列の分散はほぼ一定だからである．だから，スカートの直径の時系列では，階差を 2 回取って定常化する必要がある．

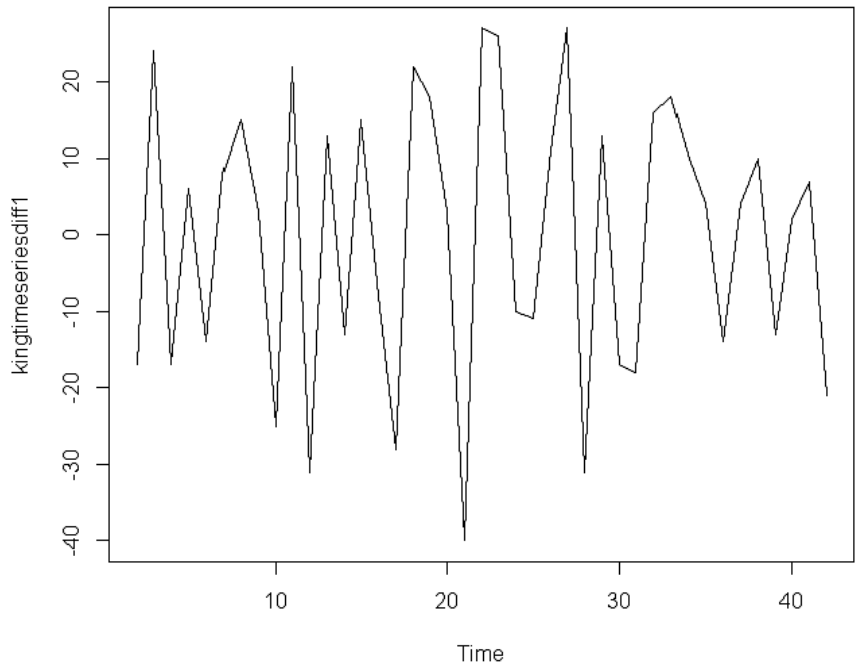
定常時系列を得るために d 回階差を取る必要がある場合，モデルとして ARIMA(p,d,q) を利用すべきであることを意味する（d は階差を取る回数）．例えば，女性のスカートの裾の直径データにおいて，階差を 2 回取ったので，$d = 2$ である．よって，この時系列に対して ARIMA(p,2,q) を利用できる．次の手順は，ARIMA モデルの p と q の値を求める．

他の例として，イングランドの歴代の王の享年の時系列を取り上げる．

推移プロットから，時系列の平均は定常ではない．階差を 1 回取り，それをプロットするには次を入力する．

```
> kingtimeseriesdiff1 <- diff(kingstimeseries, differences=1)
> plot.ts(kingtimeseriesdiff1)
```





階差を 1 回取った時系列の平均と分散は定常のようなので，イングランドの歴代の王の享年の時系列には ARIMA(p,1,q) モデルが適切のようであり，このとき残っているのは不規則成分のみである．この不規則成分に系列相関があるかどうかを確認する．もしあるなら，王の享年の予測モデルを作成するのに利用できる．

2.6.2 ARIMA モデルの候補の選択

時系列が定常であるか，それの階差を d 回取ることにより定常時系列に変換できる場合，次にすべきことは，適切な ARIMA モデルを選択することである．これは，ARIMA(p,d,q) モデルに対する p と q の最も適切な値を見つけることを意味する．これを行うには，定常時系列のコレログラムおよび偏コ

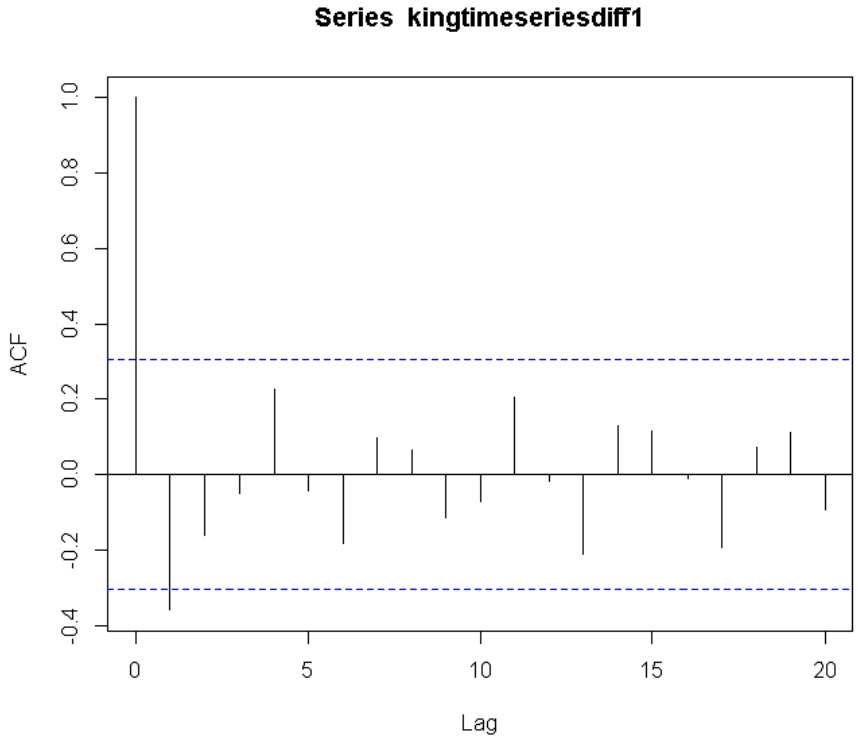
レログラムを調べるのが通常の手順である．

コレログラムと偏コレログラムのプロットには，関数 “acf()” と “pacf()” をそれぞれ利用する．自己相関係数と辺自己相関係数の値を実際に得るには，“acf()” と “pacf()” の中で “plot=FALSE” というオプションを利用する．

イングランドの歴代の王の享年の例

例えば，イングランドの歴代の王の享年の 1 回の階差のラグ $1-20$ に対するコレログラムをプロットし，自己相関の値を得るには，次を入力する．

```
> acf(kingtimeseriesdiff1, lag.max=20) # コレログラムのプロット
> acf(kingtimeseriesdiff1, lag.max=20, plot=FALSE) # 自己相関係数の取得

Autocorrelations of series ‘kingtimeseriesdiff1’, by lag

     0      1      2      3      4      5      6      7      8      9     10     11     12     13
 1.000 -0.360 -0.162 -0.050  0.227 -0.042 -0.181  0.095  0.064 -0.116 -0.071  0.206 -0.017 -0.212
    14     15     16     17     18     19     20
 0.130  0.114 -0.009 -0.192  0.072  0.113 -0.093
```

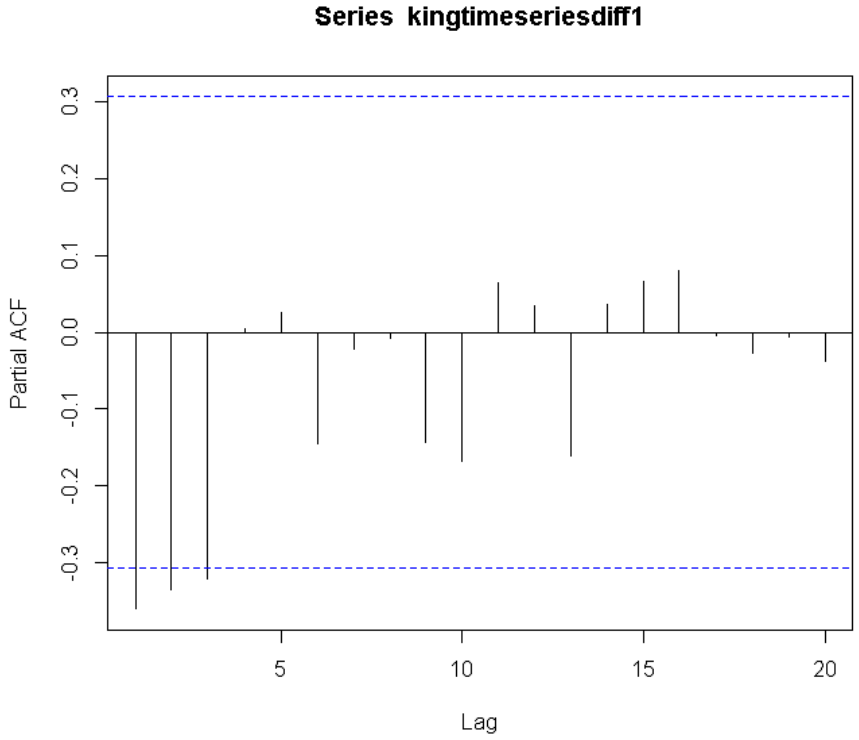


ラグ 1 における自己相関の値（-0.360）が有意となっているが，他のラグにおける自己相関は有意でないことがわかる．

王の享年データの 1 回の階差のラグ $1-20$ の偏コレログラムをプロットし，偏自己相関の値を得るには “pacf()” 関数を用いて，次を入力する．

```
> pacf(kingtimeseriesdiff1, lag.max=20) # partial correlogram のプロット
> pacf(kingtimeseriesdiff1, lag.max=20, plot=FALSE) # 偏自己相関の取得

Partial autocorrelations of series ‘kingtimeseriesdiff1’, by lag

     1      2      3      4      5      6      7      8      9     10     11     12     13     14
```

```
-0.360 -0.335 -0.321  0.005  0.025 -0.144 -0.022 -0.007 -0.143 -0.167  0.065  0.034 -0.161  0.036
    15     16     17     18     19     20
 0.066  0.081 -0.005 -0.027 -0.006 -0.037
```



偏コレログラムより，ラグ 1, 2, 3 における偏自己相関は有意で，負の値である．また，ラグが増加するにつれて，絶対値が次第に減少している (ラグ 1 : −0.360，ラグ 2 : −0.335，ラグ 3 : −0.321) ことがわかる．

ラグ 1 の後，コレログラムは 0 であり，偏コレログラムはラグ 3 以降，0 に漸減している．これは，1 回の階差の時系列において平均が次に示す 3 つの ARMA（自己回帰移動平均 : autoregressive moving average）モデルのいずれかに従っている可能性を意味する．

- ARMA(3,0) モデル（次数 $p = 3$ の自己回帰モデル）．ラグ 3 以降，偏コレログラムは 0 となり，コレログラムは 0 に漸減しているから（このモデルが適切であるためには，コレログラムは急激に 0 に減少する必要があるが）．
- ARMA(0,1) モデル（次数 $q = 1$ の移動平均モデル）．ラグ 1 以降コレログラムは 0 となり，偏コレログラムは 0 に漸減しているから．
- ARMA(p,q) モデル（次数 p，q（> 0）の混合モデル）．コレログラムと偏コレログラムは 0 に漸減しているから（このモデルが適切であるには，コレログラムは急激に 0 に減少する必要があるが）．

どのモデルが適切かを決定するために “けちの原理” を利用する．すなわち，パラメータが最小のモデルが最適であると仮定する．ARMA(3,0) モデルは 3 つのパラメータを，ARMA(0,1) は 1 つのパラメータを，ARMA(p,q) モデルは少なくとも 2 つのパラメータを持つ．よって，ARMA(0,1) モデルを最適モデルとする．

ARMA(0,1) は，次数 1 の移動平均モデルであり，MA(1) モデルともいう．このモデルの式は，$X_t - mu = Z_t - (theta * Z_t - 1)$ と書くことができる．ここで，X_t は分析している定常時系列であり (イングランドの王の享年の 1 回の階差)，mu は時系列 X_t の平均，Z_t は平均 0，分散が一定のホワイトノイズ，$theta$ は推定することができるパラメータである．

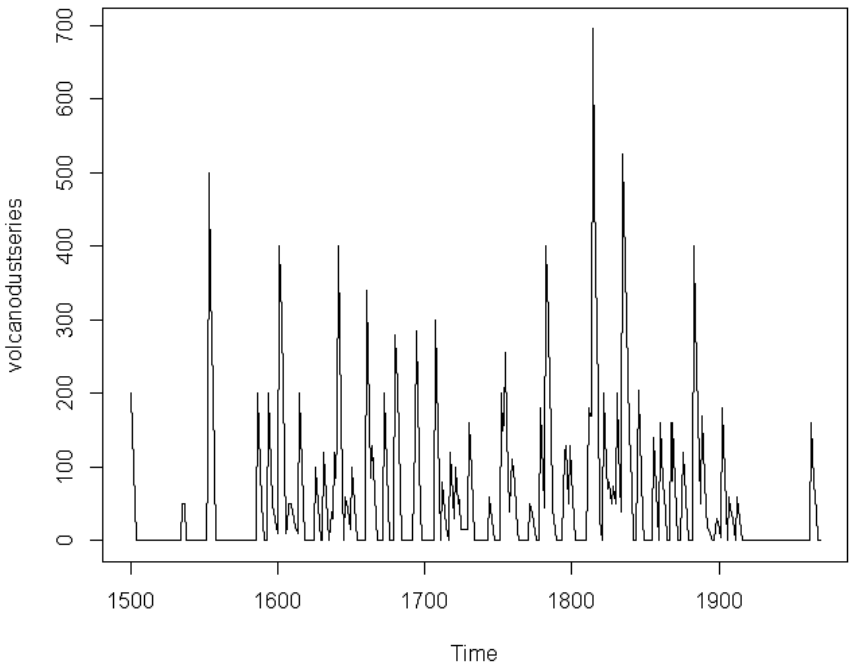
MA（移動平均）モデルは，通常，連続する観測値の短期の従属性を示す時系列のモデル化に用いられる．イングランドの王の享年の時系列において，不規則成分を記述するのに MA モデル用いることができることは直感的にわかる．なぜなら，特定の王の享年が次の，またはその次の王の享年に影響を与える可能性は高いが，はるかに後代の王の享年には影響を与えないと考えるのが自然であるからである．

イングランドの王の享年の 1 回の階差に対して，ARMA(0,1) モデル（$p = 0$，$q = 1$）が最適なモデルの候補なので，この時系列は，ARIMA(0,1,1) モデル（$p = 0, d = 1, q = 1$ で，d は階差を取る回数）を用いてモデル化することができる．

北半球の火山の噴煙の例

適切な ARIMA モデルを選択する別の例を取り上げる．ファイル `http://robjhyndman.com/tsdldata/annual/dvi.dat` には, 1500 年から 1969 年までの北半球における火山の噴煙指数（volcanic dust veil index）のデータがある（Hipel and Mcleod, 1994 より）．これは火山の噴火による噴煙とエアロゾルが環境に与える影響の指標である．これを R に読み込み，推移グラフを作成するには，次を入力する．

```
> volcanodust <- scan("http://robjhyndman.com/tsdldata/annual/dvi.dat", skip=1)
Read 470 items
> volcanodustseries <- ts(volcanodust,start=c(1500))
> plot.ts(volcanodustseries)
```



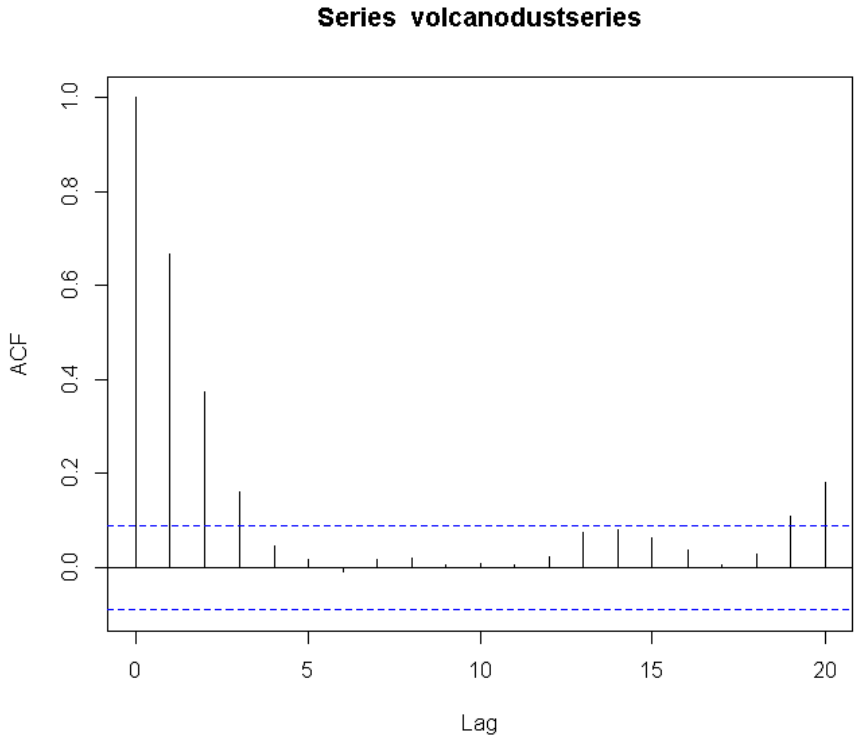
推移グラフより，時系列の不規則変動の大きさはほぼ等しいので，この時系列を記述するのに加法モデルが適切であると予想できる．

さらに，この時系列の平均と分散はほぼ一定なので，定常である．よって，原系列に ARIMA モデルを適用するために階差を取る必要はない（つまり，必要な回数 d は 0）．

どの ARIMA モデルを利用するかを調べるために，ラグ $1 - 20$ のコレログラムと偏コレログラムをプロットする．

```
> acf(volcanodustseries, lag.max=20) # コレログラムのプロット
> acf(volcanodustseries, lag.max=20, plot=FALSE) # 自己相関係数の取得
```

```
Autocorrelations of series ‘volcanodustseries’, by lag

    0     1     2     3     4     5     6     7     8     9    10    11    12    13
1.000 0.666 0.374 0.162 0.046 0.017 -0.007 0.016 0.021 0.006 0.010 0.004 0.024 0.075
   14    15    16    17    18    19    20
0.082 0.064 0.039 0.005 0.028 0.108 0.182
```



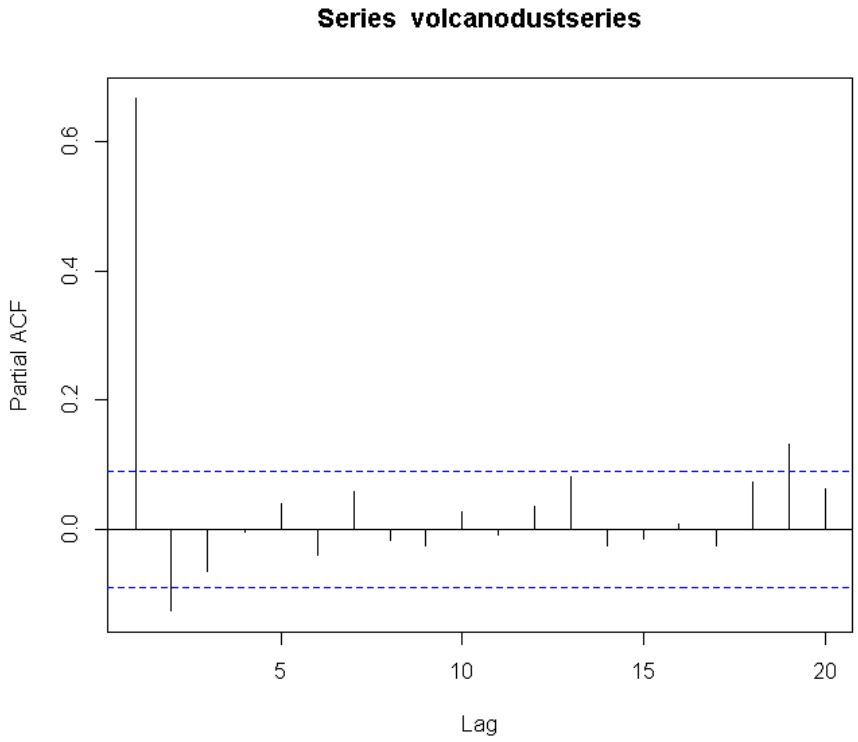
コレログラムより，ラグ 1, 2, 3 に対する自己相関は有意性の限界を超えており，ラグ 3 以降，自己相関は 0 に収束していることがわかる．ラグ 1, 2, 3 に対する自己相関は正であり，ラグが大きくなるにつれて小さくなっている（ラグ 1 のとき 0.666，ラグ 2 のとき 0.374，ラグ 3 のとき 0.162）．

ラグ 19, 20 の自己相関も有意性の限界を超えているが，これは偶然の可能性が高い．なぜなら，ほぼ限界値に近いところにあり（特にラグ 19 の場合），ラグ $4-18$ の自己相関は超えておらず，20 個のラグのうち 1 つが 95% 有意性の限界を偶然に超える可能性はあるからである．

```
> pacf(volcanodustseries, lag.max=20)
> pacf(volcanodustseries, lag.max=20, plot=FALSE)

Partial autocorrelations of series ‘volcanodustseries’, by lag

    1     2     3     4     5     6     7     8     9    10    11    12    13    14
0.666 -0.126 -0.064 -0.005  0.040 -0.039  0.058 -0.016 -0.025  0.028 -0.008  0.036  0.082 -0.025
   15    16    17    18    19    20
-0.014  0.008 -0.025  0.073  0.131  0.063
```

偏自己コレログラムより，ラグ 1 の偏自己相関（0.666）は正であり，有意性の限界を超えており，ラグ 2 の偏自己相関は負（-0.126）であり，有意性の限界を超えている．ラグ 3 以降，偏自己相関は 0 に漸減している．

ラグ 3 以降，コレログラムは 0 に漸減しているので，この時系列に対して次に示す 3 つの ARMA モデルを適用することが可能である．



- ARMA(2,0) モデル（次数 $p = 2$ の自己回帰モデル）．なぜなら，ラグ 2 以降，偏自己相関は 0 となり，ラグ 3 以降，コレログラムは 0 に収束しているから（このモデルが適切であるためには，コレログラムが急激に 0 に収束する必要があるが）．
- ARMA(0,3) モデル．ラグ 3 以降自己相関は 0 となり，偏自己相関は 0 に収束しているから（このモデルが適切であるためには，急速に 0 に収束する必要はあるが）．
- ARMA(p,q) 混合モデル．コレログラムと偏コレログラムが 0 に漸減しているから（このモデルが適切であるには，コレログラムが急激に 0 に収束する必要はあるが）．

ARMA(2,0) モデルは 2 つのパラメータを，ARMA(0,3) は 3 つのパラメータを，ARMA(p,q) モデルは少なくとも 2 つのパラメータを持つ．よって，けちの原理に従うと，ARMA(2,0) モデルと ARMA(p,q) モデルが候補のモデルとなる．

ARMA(2,0) は，次数 2 の移動平均モデルであり，MA(2) モデルともいう．このモデルの式は，$X_t - mu = (Beta1 * (X_{t-1} - mu)) + (Beta2 * (X_{t-2} - mu)) + Z_t$ となる．ここで，X_t は分析している定常時系列であり（火山の噴煙指数の時系列），mu は時系列 X_t の平均，$Beta1$ と $Beta2$ は推定すべきパラメータ，Z_t は平均 0，分散が一定のホワイトノイズである．

AR（自己回帰）モデルは，通常，連続する観測値の長期の従属性を示す時系列のモデル化に用いられる．火山の噴煙指数の時系列を記述するのに AR モデル用いることができることは直感的にわかる．ある年の火山の噴煙とエアロゾルの水準は，長い期間にわたって影響を与える可能性が高いからである．なぜなら，噴煙やエアロゾルが素早く消えることは期待できない．

火山の噴煙指数の時系列に対して，ARMA(2,0) モデル（$p = 2$，$q = 0$）を用いるとき，この時系列に，ARIMA(2,0,0) モデル（$p = 2$，$d = 0$，$q = 0$ で，d は階差を取るべき回数）を適用できる．同様に，ARMA(p,q) 混合モデル（p および q は正の値）の場合，ARIMA(p,0,q) モデルを適用できる．

2.6.3 ARIMA モデルを用いた予測

時系列データに対して最適な ARIMA(p,d,q) モデルの候補を選択した後，ARIMA モデルのパラメータを推定し，それを予測モデルとして利用することができる．

ARIMA(p,d,q) モデルのパラメータを推定するには，“arima()” 関数を用いることができる．

イングランドの王の享年の例

ARIMA(0,1,1) モデルはイングランドの王の享年に対する妥当なモデルのようである．“arima()” 関数では，“order” 引数により，ARIMA モデルの p, d, q の値を指定することができる．ARIMA(p,d,q) モデルをこの時系列（変数名 “*kingstimeseries*” として保存している）にあてはめるには，次を入力する．

```
> kingstimeseriesarima <- arima(kingstimeseries, order=c(0,1,1)) # fit an ARIMA(0,1,1) model
> kingstimeseriesarima
Series: kingstimeseries
ARIMA(0,1,1)

Coefficients:
          ma1
      -0.7218
s.e.   0.1208

sigma^2 estimated as 230.4:  log likelihood=-170.06
AIC=344.13   AICc=344.44   BIC=347.56
```

既に述べたように, ARIMA(0,1,1) モデルをこのデータに当てはめるには, 階差を 1 回取った時系列に ARMA(0,1) モデルを当てはめればよい．ARMA(0,1) モデルの式は，$X_t - mu = Z_t - (theta * Z_t - 1)$, ここで $theta$ は推定するパラメータ，と書くことができる．“arima()” 関数の出力から，$theta$ の推定値（R の出力中の ‘mal’ に保存）は，ARIMA(0,1,1) モデルの場合，-0.7218 となる．

次に，ARIMA モデルを利用してこの時系列の将来の予測を行うことができる．利用する関数は “forecast” パッケージにある “forecast.Arima()” 関数である．例えば，イングランドの次の 5 代の王の享年を予測するには，次を入力する．

```
> library("forecast") # "forecast"ライブラリのロード
> kingstimeseriesforecasts <- forecast.Arima(kingstimeseriesarima, h=5)
> kingstimeseriesforecasts
   Point Forecast    Lo 80    Hi 80    Lo 95     Hi 95
43       67.75063 48.29647 87.20479 37.99806  97.50319
44       67.75063 47.55748 87.94377 36.86788  98.63338
45       67.75063 46.84460 88.65665 35.77762  99.72363
46       67.75063 46.15524 89.34601 34.72333 100.77792
47       67.75063 45.48722 90.01404 33.70168 101.79958
```

英国の王の原系列は，42 代の王の享年からなる．forecast.Arima() 関数は，次の 5 代（$43-47$ 代）の享年の予測値のみならず，80% および 95% の予測区間も与える．42 代の王の享年は 56 歳（時系列の最後のデータ）であるが，ARIMA モデルによる次の 5 代の王の予測された享年は 67.8 歳である．

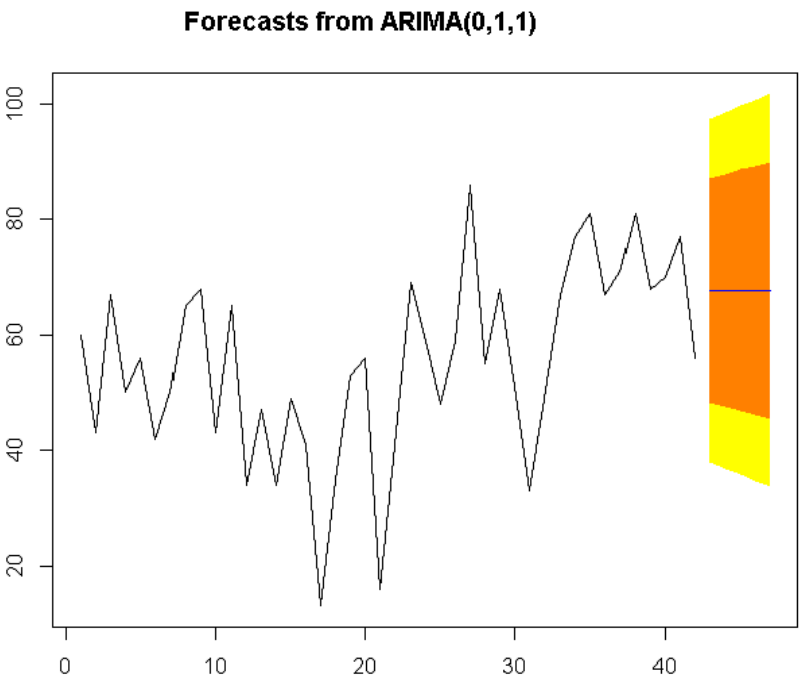
最初の 42 代の王の享年の観測値をプロットするとともに，ARIMA(0,1,1) を用いて次の 5 代の王の享年の予測値をプロットするには，次を入力する．

```
> plot.forecast(kingstimeseriesforecasts)
```

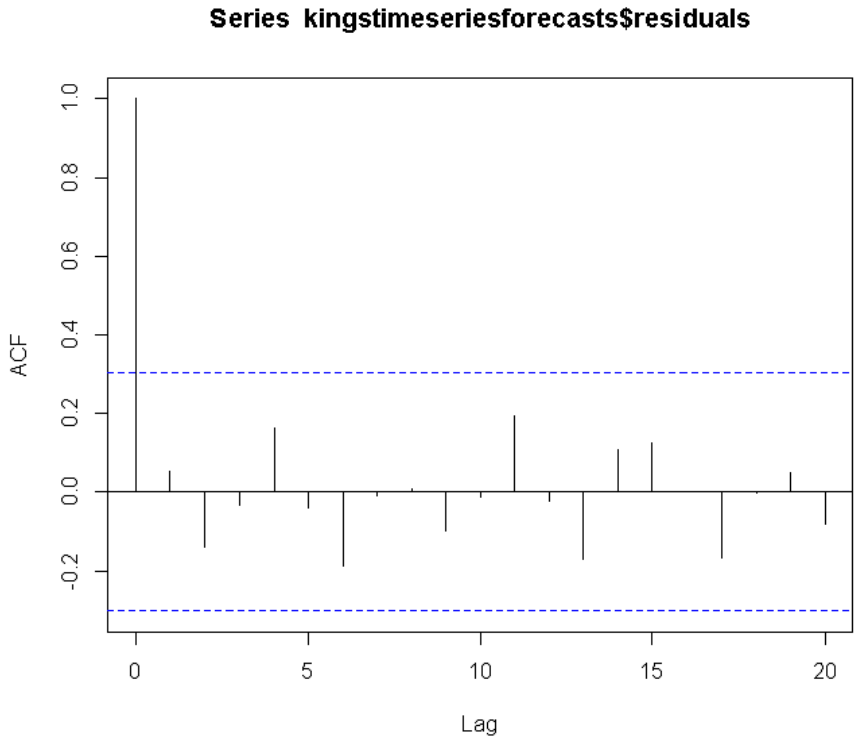
指数平滑法と同様，ARIMA モデルの予測誤差が平均が 0，分散が一定の正規分布に従っているかどうか，予測誤差に系列相関があるかどうかを確認することを推奨する．

例えば，王の享年に対して ARIMA(0,1,1) に対する予測誤差のコレログラムを作成し，ラグが $1-20$ に対する Ljung-Box 検定を行うには，次を入力する．

```
> acf(kingstimeseriesforecasts$residuals, lag.max=20)
> Box.test(kingstimeseriesforecasts$residuals, lag=20, type="Ljung-Box")
```




```
        Box-Ljung test

data:  kingstimeseriesforecasts$residuals
X-squared = 13.5844, df = 20, p-value = 0.8509
```

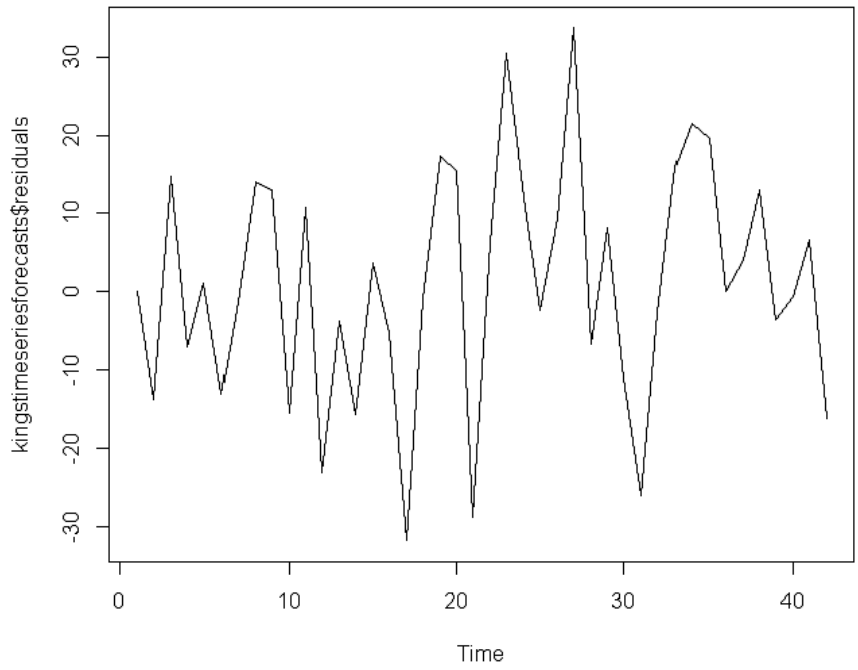


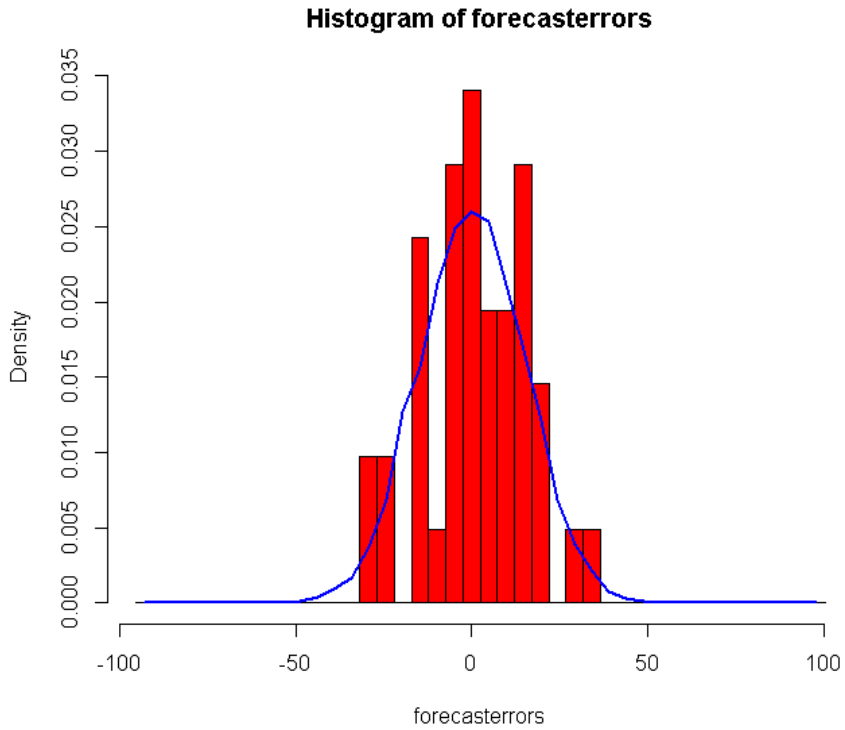
ラグ 1 – 20 に対するコレログラムはどれも有意性の限界を超えておらず，Ljung-Box 検定の p 値は

0.9 なので，ラグ 1 – 20 に対する予測誤差に自己相関があるという証拠はほとんどない．

予測誤差の分布が，平均 0，一定の分散の正規分かどうかを確認するには，予測誤差の推移グラフとヒストグラム（正規分布を重ねて描いて）を作成すればよい．

```
> plot.ts(kingstimeseriesforecasts$residuals) # 予測誤差の推移グラフ
> plotForecastErrors(kingstimeseriesforecasts$residuals) # ヒストグラムの作成
```





標本内予測誤差の推移グラフより，予測誤差の分散はほぼ一定である（時系列の前半と比べ，後半で分散が少し大きくなっているようであるが）．ヒストグラムより，予測誤差の分布はほぼ正規分布であ

り，平均は 0 に近いことが分かる．よって，予測誤差の分布は，平均が 0，分散が一定の正規分布であるというのは妥当な判断である．

予測誤差には系列相関がないようであり，予測誤差の分布が平均 0，分散が一定の正規分布のようなので，ARIMA(0,1,1) モデルはイングランドの王の享年に対する適切な予測モデルであると判断できる．

北半球の火山の噴煙の例

火山の噴煙指数に対する適切な ARIMA モデルは，ARIMA(2,0,0) であることを議論した．この時系列に ARIMA(2,0,0) をあてはめるには，次を入力する．

```
> volcanodustseriesarima <- arima(volcanodustseries, order=c(2,0,0))
> volcanodustseriesarima
Series: volcanodustseries
ARIMA(2,0,0) with non-zero mean

Coefficients:
         ar1      ar2  intercept
      0.7533  -0.1268    57.5274
s.e.  0.0457   0.0458     8.5958

sigma^2 estimated as 4870:  log likelihood=-2662.54
AIC=5333.09   AICc=5333.17   BIC=5349.7
```

既に述べたように，ARIMA(2,0,0) モデルの式は，$X_t - mu = (Beta1 * (X_{t-1} - mu)) + (Beta2 * (X_{t-2} - mu)) + Z_t$ である．ここで，$Beta1$ と $Beta2$ は推定すべきパラメータである．arima() 関数の出力より，$Beta1$ の推定値は 0.7533，$Beta2$ の推定値は -0.1268 である（arima() の出力の ar1，ar2 の値）．

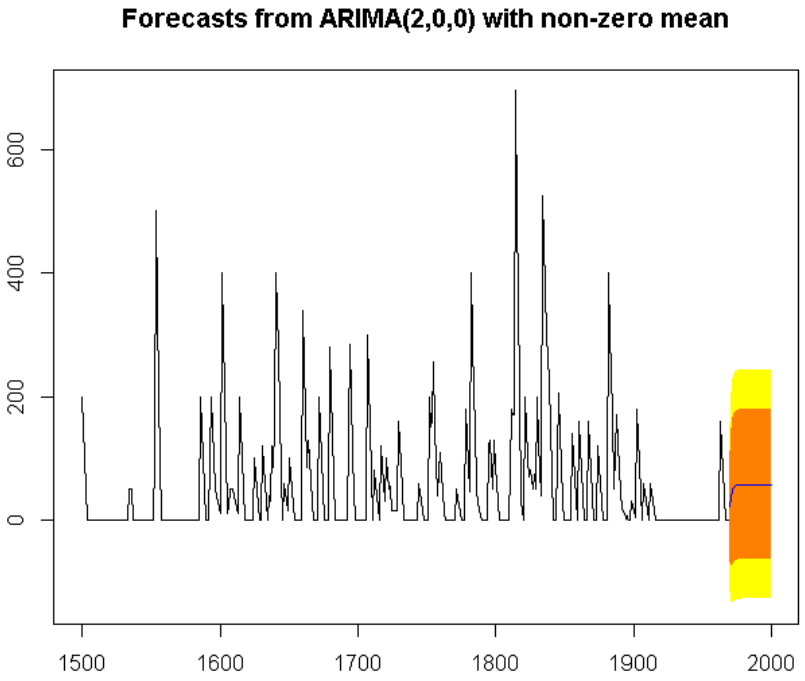
ARIMA(2,0,0) モデルに当てはめると，“forecast.ARIMA()” を用いて火山の将来の噴煙指数を予測することができる．原系列は 1500 年から 1969 年までであった．1970 年から 2000 年（31 年分）までの予測を行うには，次を入力する．

```
> volcanodustseriesarima <- arima(volcanodustseries, order=c(2,0,0))
> volcanodustseriesarima
Series: volcanodustseries
ARIMA(2,0,0) with non-zero mean

Coefficients:
         ar1      ar2  intercept
      0.7533  -0.1268    57.5274
s.e.  0.0457   0.0458     8.5958

sigma^2 estimated as 4870:  log likelihood=-2662.54
AIC=5333.09   AICc=5333.17   BIC=5349.7
> volcanodustseriesforecasts <- forecast.Arima(volcanodustseriesarima, h=31)
> volcanodustseriesforecasts
     Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
1970       21.48131 -67.94860 110.9112 -115.2899 158.2526
1971       37.66419 -74.30305 149.6314 -133.5749 208.9033
1972       47.13261 -71.57070 165.8359 -134.4084 228.6737
1973       52.21432 -68.35951 172.7881 -132.1874 236.6161
1974       54.84241 -66.22681 175.9116 -130.3170 240.0018
1975       56.17814 -65.01872 177.3750 -129.1765 241.5327
1976       56.85128 -64.37798 178.0805 -128.5529 242.2554
1977       57.18907 -64.04834 178.4265 -128.2276 242.6057
```

```
1978       57.35822 -63.88124 178.5977 -128.0615 242.7780
1979       57.44283 -63.79714 178.6828 -127.9777 242.8634
1980       57.48513 -63.75497 178.7252 -127.9356 242.9059
1981       57.50627 -63.73386 178.7464 -127.9145 242.9271
1982       57.51684 -63.72330 178.7570 -127.9040 242.9376
1983       57.52212 -63.71802 178.7623 -127.8987 242.9429
1984       57.52476 -63.71538 178.7649 -127.8960 242.9456
1985       57.52607 -63.71407 178.7662 -127.8947 242.9469
1986       57.52673 -63.71341 178.7669 -127.8941 242.9475
1987       57.52706 -63.71308 178.7672 -127.8937 242.9479
1988       57.52723 -63.71291 178.7674 -127.8936 242.9480
1989       57.52731 -63.71283 178.7674 -127.8935 242.9481
1990       57.52735 -63.71279 178.7675 -127.8934 242.9481
1991       57.52737 -63.71277 178.7675 -127.8934 242.9482
1992       57.52738 -63.71276 178.7675 -127.8934 242.9482
1993       57.52739 -63.71275 178.7675 -127.8934 242.9482
1994       57.52739 -63.71275 178.7675 -127.8934 242.9482
1995       57.52739 -63.71275 178.7675 -127.8934 242.9482
1996       57.52739 -63.71275 178.7675 -127.8934 242.9482
1997       57.52739 -63.71275 178.7675 -127.8934 242.9482
1998       57.52739 -63.71275 178.7675 -127.8934 242.9482
1999       57.52739 -63.71275 178.7675 -127.8934 242.9482
2000       57.52739 -63.71275 178.7675 -127.8934 242.9482
```

原系列と予測値とをプロットするには，次を入力する．

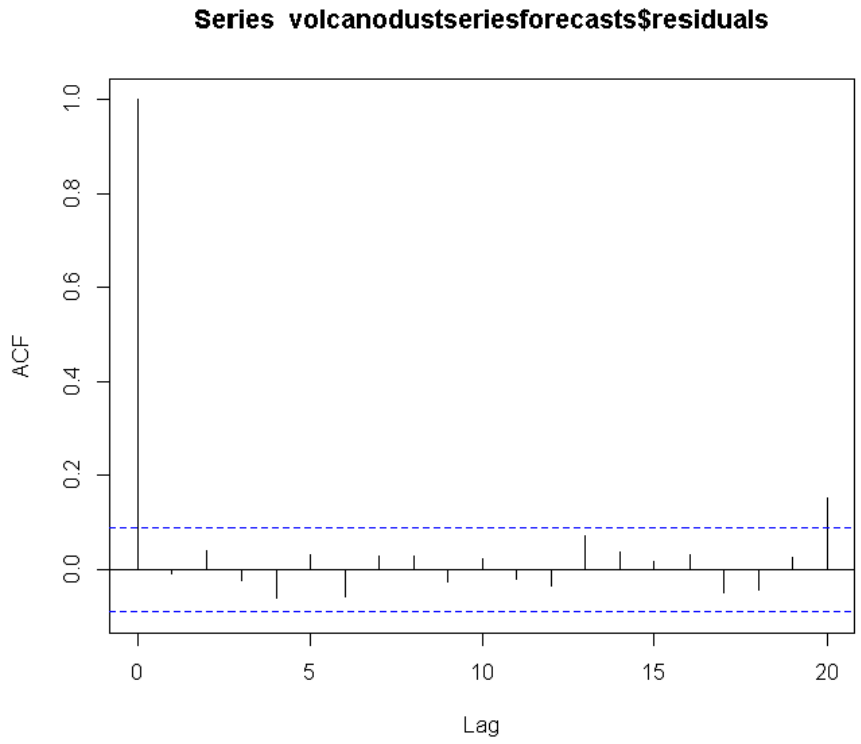
```
> plot.forecast(volcanodustseriesforecasts)
```



1 つ気になるところは，火山の噴煙指数は正の値であるにもかかわらず，予測値が負になっていることである．理由は，arima() と forecast.Arima() 関数が正の値しか取らない変数であることを知らないからである．明らかに，これは現在の予測モデルに対するあまり望ましくない特徴である．

ここでもまた，予測誤差が相関を持つかどうか，平均が 0，分散が一定の正規分布に従うかどうかを調べる必要がある．予測誤差の系列相関があるかどうかを確認するには，コレログラムを作成し，Ljung-Box 検定を行えばよい．

```
> acf(volcanodustseriesforecasts$residuals, lag.max=20)
> Box.test(volcanodustseriesforecasts$residuals, lag=20, type="Ljung-Box")

        Box-Ljung test

data:  volcanodustseriesforecasts$residuals
X-squared = 24.3642, df = 20, p-value = 0.2268
```



コレログラムより，ラグ 20 での標本自己相関が有意性の限界を超えていることがわかる．しかし，これは単なる偶然であろう．なぜなら，20 個の標本の中で 1 つが 95% 信頼限界を超える可能性はあるからである．さらに，Ljung-Box 検定の p 値は 0.2 であり，これは，ラグ $1-20$ に対する予測誤差に自己相関があるという証拠にはならない．

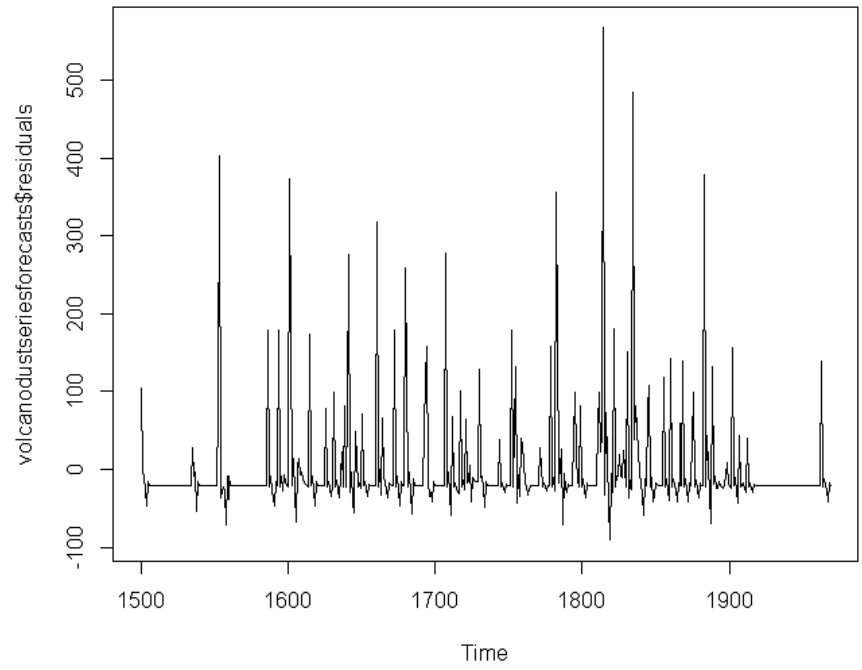
予測誤差の分布が，平均が 0，分散が一定の正規分布に従っているかどうかの検証には，予測誤差の推移グラフとヒストグラムを作成する．

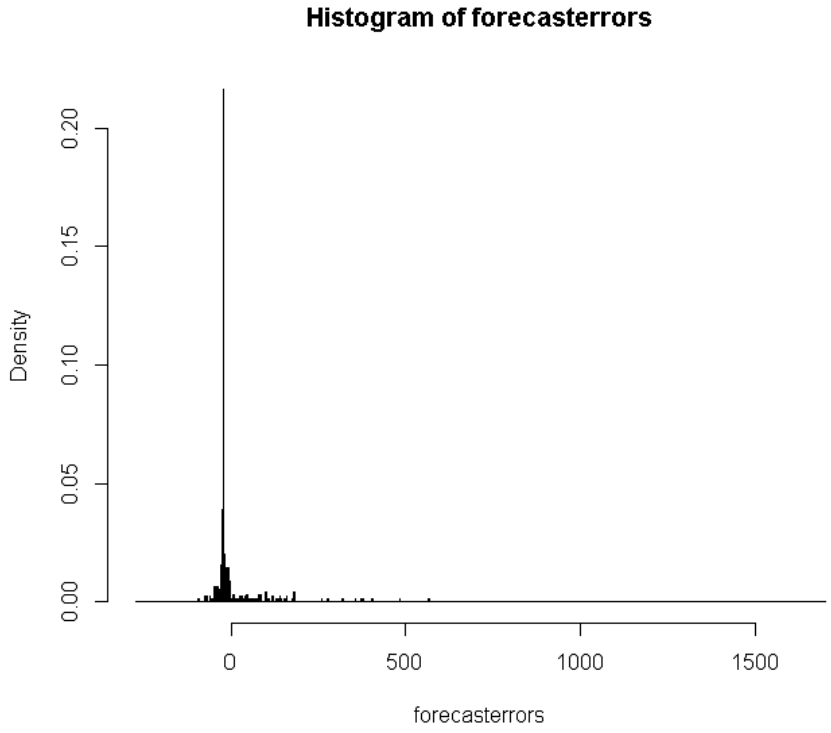
```
> plot.ts(volcanodustseriesforecasts$residuals) # 予測誤差の推移グラフ
> plotForecastErrors(kingstimeseriesforecasts$residuals) # ヒストグラムの作成
```

予測誤差の推移図より，予測誤差はほぼランダムに変動していることがわかる．しかし，予測誤差の時系列の平均は，0 ではなく負となっている．このことは，予測誤差の平均を計算することにより (-0.22 となる)，確認することができる．

```
> mean(volcanodustseriesforecasts$residuals)
[1] -0.2205417
```

予測誤差のヒストグラムより，正規分布と比べて右に歪んでいることがわかる．よって，予測誤差の分布が平均が 0，分散が一定の正規分布であると判断することは難しい．だから，火山の噴煙指数に対





する作成した ARIMA(2,0,0) モデルは，最良のモデルではなく，もっと改良できそうである．

2.7　リンクと参考文献

さらに学習するためのリンク・参考文献を紹介する．

R をより詳細に知るには，“Kickstarting R” ウェブサイト（cran.rproject.org/doc/contrib/Lemon-kickstart）にある良いオンライン・チュートリアルを参照のこと．

別のチュートリアルとしてもう少し詳細なものが，“Introduction to R” ウェブサイト（cran.

rproject.org/doc/manuals/R-intro.html）にある．

時系列解析を学ぶには，Open University 発行の本 “Time Series”（製品コード M249/02．Open University のショップより購入可）を強く推薦する．

時系列解析のための R の使用法に関する本が，“Use R!” シリーズに 2 冊ある．1 つは Cowpertwait and Metcalfe 著の “Introductory Time Series with R” であり，もう 1 つは Pfaff 著の “Analysis of Integrated and Cointegrated Time Series with R” である．

2.8 謝辞

時系列データライブラリ（Time Series Data Library : TSDL）のデータを本ブックレットで利用することを許可してくださった Rob Hyndman 教授に感謝する．

本ブックレットの例の多くは，Open University のショップで入手できる優れた本，“Time Series”（製品コード M249/02）より着想を得た．

2.9 連絡

本書に関して間違いや提案があれば，私（Avril Coghlan）にメール（アドレス a.coghlan@ucc.ie）を送付していただけると幸いである．

2.10 ライセンス

第 3 章

謝辞

本書を作成する際の Sphinx（http://sphinx.pocoo.org），異なるバージョンの管理のための github（https://github.com/），ビルドし，ディストリビュートするための readthedocs（http://readthedocs.org/）の使い方について支援を受けたことに対して，Noel O'Boyle に感謝する．

第 4 章

連絡

本書に関して間違いや提案があれば私（Avril Coghlan）にメール（a.coghlan@ucc.ie）を送付してもらえれば幸いである.

第 5 章

ライセンス

本書の内容は，クリエイティブ・コモンズ（Creative Commons）Attribution 3.0 ライセンスで公開している