1. Introduction to time series analysis

Edward Ionides

1/03/2018

Contents

Objectives for this chapter

1. Discuss some basic motivations for the topic of time series analysis.
2. Introduce some fundamental concepts for time series analysis: stationarity, autocorrelation, autoregressive models, moving average models, autoregressive-moving average (ARMA) models, state-space models. These will be covered in more detail later.
3. Develop the computational framework for this course:
    - R and Rmarkdown for data analysis and reproducible documents
    - Source code sharing using git.

1.1 Overview

- Time series data are, simply, data collected at many different times.
- This is a common type of data! Observations at similar time points are often more similar than more distant observations.
- This immediately forces us to think beyond the independent, identically distributed assumptions fundamental to much basic statistical theory and practice.
- Time series dependence is an introduction to more complicated dependence structures: space, space/time, networks (social/economic/communication), . . .

1.1.1 Looking for trends and relationships in dependent data

- The first half of this course focuses on:
    1. Quantifying dependence in time series data.

2. Finding statistical arguments for the presence or absence of associations that are valid in situations with dependence.

- Example questions: Does Michigan show evidence for global warming? Does Michigan follow global trends, or is there evidence for regional variation? What is a good prediction interval for weather in the next year or two?

1.1.2 Modeling and statistical inference for dynamic systems

- The second half of this course focuses on:
  1. Building models for dynamic systems, which may or may not be linear and Gaussian.
  2. Using time series data to carry out statistical inference on these models.
- Example questions: Can we develop a better model for understanding variability of financial markets (known in finance as volatility)? How do we assess our model and decide whether it is indeed an improvement?

1.2 A simple example: Winter in Michigan

The previous winter was mild by Michigan standards. What should we expect this year? Is there a noticeable trend? Let's look at some data.

I downloaded from www.usclimatedata.com and put in ann_arbor_weather.csv.

- You can get this file from the course website (http://ionides.github.io/531w18).
- Better, you can set up a local git repository that will give you an up-to-date copy of all the data, notes, code, homeworks and solutions for this course. More on this later.

```
y <- read.table(file="ann_arbor_weather.csv",header=1)
```

Here, I'm using R Markdown to combine source code with text. This gives a nice way to generate statistical analysis that is

1. Reproducible,
2. Easily modified or extended.

These two properties are useful for developing your own statistical research projects. Also, they are useful for teaching and learning statistical methodology, since they make it easy for you to replicate and adapt analysis presented in class.

1.2.1 Question: How many of you already know R Markdown?

First, let's get some basic idea what's in our dataset. `str` summarizes the structure of the data:

```r
str(y)
```

```
## 'data.frame':    118 obs. of  12 variables:
##  $ Year     : int  1900 1901 1902 1903 1904 1905 1906 1907 1908 1909 ...
##  $ Low      : num  -7 -7 -4 -7 -11 -3 11 -8 -8 -1 ...
##  $ High     : num  50 48 41 50 38 47 62 61 42 61 ...
##  $ Hi_min   : num  36 37 27 36 31 32 53 38 32 50 ...
##  $ Lo_max   : num  12 20 11 12 6 14 20 11 15 13 ...
##  $ Avg_min  : num  18 17 15 15.1 8.2 10.9 25.8 17.2 17.6 20 ...
##  $ Avg_max  : num  34.7 31.8 30.4 29.6 22.9 25.9 38.8 31.8 28.9 34.7 ...
##  $ Mean     : num  26.3 24.4 22.7 22.4 15.3 18.4 32.3 24.5 23.2 27.4 ...
##  $ Precip   : num  1.06 1.45 0.6 1.27 2.51 1.64 1.91 4.68 1.06 2.5 ...
##  $ Snow     : num  4 10.1 6 7.3 11 7.9 3.6 16.1 4.3 8.7 ...
##  $ Hi_Pricip: num  0.28 0.4 0.25 0.4 0.67 0.84 0.43 1.27 0.63 1.27 ...
##  $ Hi_Snow  : num  1.1 3.2 2.5 3.2 2.1 2.5 2 5 1.3 7 ...
```

Let's focus on `Low`, which is the lowest temperature, in Fahrenheit, for January of each year.

- There are practical reasons to understand the expected (i.e., mean) low temperature and the annual variation around this. Reasons to do this could be
  - Agriculture: can I grow ginseng in Ann Arbor?
  - Energy: assess the cost-effectiveness of installing extra home insulation.
  - Lifestyle: Should I move to Minneapolis, or Berlin, or Beijing?
- Also, we could develop our analysis to look for evidence of climate change.
- As statisticians, we want an uncertainty estimate. We want to know how reliable our estimate is, since it is based on only a limited amount of data.
- The data are $y_1^*, \dots, y_N^*$, which we also write as $y_{1:N}^*$.
- Basic estimates of the mean and standard deviation are

$$\hat{\mu}_1^* = \frac{1}{N}\sum_{n=1}^{N} y_n^*, \qquad\qquad \hat{\sigma}_1^* = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(y_n^* - \hat{\mu}_1)^2}.$$

- We are being pedantic about adding the asterisks to denote that $\hat{\mu}_1^*$ is the **estimate** derived from evaluating the **estimator** $\hat{\mu}_1$ using the data $y_{1:N}^*$.
- This suggests an approximate confidence interval for μ of $\hat{\mu}_1^* \pm 1.96\,\hat{\sigma}_1^*/\sqrt{N}$,
- 1955 has missing data, coded as `NA`, requiring a minor modification. So, we compute $\hat{\mu}_1^*$ and $\mathrm{SE}_1 = \hat{\sigma}_1^*/\sqrt{N}$ as

```r
mu1 <- mean(y$Low,na.rm=TRUE)
se1 <- sd(y$Low,na.rm=TRUE)/sqrt(sum(!is.na(y$Low)))
cat("mu1 =", mu1, ",  se1 =", se1, "\n")
```

```
## mu1 = -2.717949 ,  se1 = 0.675042
```

- Note that SE_1 is also an estimate. We could therefore write SE_1^*, but here there is less room for misinterpretation.

1.2.2 Question: What are the assumptions behind the resulting confidence interval, -2.72 ± 1.32.

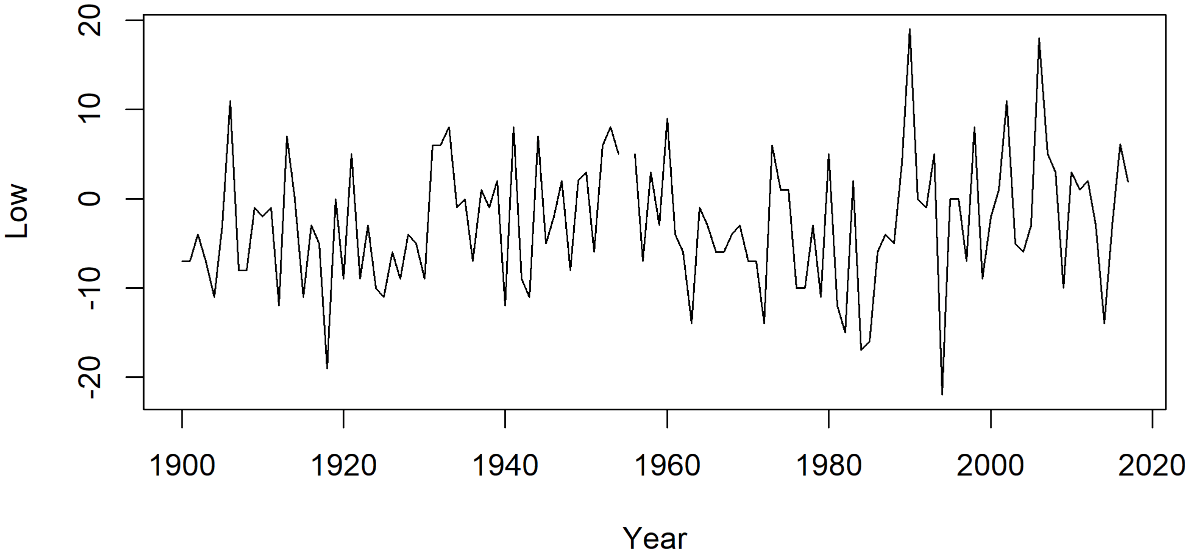
1.2.3 Question: When, in practice, is it reasonable to present this confidence interval? Is it reasonable here?

1.2.4 Question: How would you proceed?

1.2.5 Some data analysis

- The first rule of data analysis is to plot the data in as many ways as you can think of! For time series, we usually start with a time plot

```r
plot(Low~Year,data=y,ty="l")
```



- Another simple thing to do is to fit an **autoregressive-moving average** (ARMA) model. We'll look at ARMA models in much more detail later in the course.
- Let's fit an ARMA model given by

$$Y_n = \mu + \alpha(Y_{n-1} - \mu) + \epsilon_n + \beta\epsilon_{n-1}.$$

  This has a one-lag autoregressive term, $\alpha(Y_{n-1} - \mu)$, and a one-lag moving average term, $\beta\epsilon_{n-1}$. It is therefore called an ARMA(1,1) model. These lags give the model some time dependence.

  - If $\alpha = \beta = 0$, we get back to the basic independent model, $Y_n = \mu + \epsilon_n$.
  - If $\alpha = 0$ we have a moving average model with one lag, MA(1).

  - If $\beta = 0$, we have an autoregressive model with one lag, AR(1).

- We model $\epsilon_1 \dots, \epsilon_N$ to be an independent, identically distributed sequence. To be concrete, let's specify a model where they are normally distributed with mean zero and variance σ^2.

- A note on notation:

  - In this course, capital Roman letters, e.g., X, Y, Z, denote random variables. We may also use ϵ, η, ξ, ζ for random noise processes. Thus, these symbols are used to build models.

  - We use lower case Roman letters (x, y, z, $\dots$) to denote fixed numbers.

  - When fixed numbers are data, or are derived from data, we will often add an asterisk (x^*, y^*, $\dots$). A quantity derived from data is called a **statistic**.

  - "We must be careful not to confuse data with the abstractions we use to analyze them." (William James, 1842-1910).

  - Other Greek letters will usually be parameters, i.e., real numbers that form part of the model.

- We can readily fit the ARMA(1,1) model by maximum likelihood,

```
arma11 <- arima(y$Low, order=c(1,0,1))
```

We can see a summary of the fitted model, where α is called `ar1`, β is called `ma1`, and μ is called `intercept`.

```
arma11
```

```
##
## Call:
## arima(x = y$Low, order = c(1, 0, 1))
##
## Coefficients:
##          ar1      ma1  intercept
##       0.8400  -0.7972    -2.7103
## s.e.  0.2424   0.2661     0.8387
##
## sigma^2 estimated as 52.53:  log likelihood = -397.77,  aic = 803.55
```

SE's created by taking fisher information matrix at MLE

We will write the ARMA(1,1) estimate of μ as $\hat{\mu}_2^*$, and its standard error as SE_2. Some poking around is required to extract the quantities of primary interest from the fitted ARMA model in R.

```
names(arma11)
```

```
##  [1] "coef"      "sigma2"    "var.coef"  "mask"      "loglik"
##  [6] "aic"       "arma"      "residuals" "call"      "series"
## [11] "code"      "n.cond"    "nobs"      "model"
```

```
mu2 <- arma11$coef["intercept"]
se2 <- sqrt(arma11$var.coef["intercept","intercept"])
cat("mu2 =", mu2, ",  se2 =", se2, "\n")
```

```
## mu2 = -2.710296 ,  se2 = 0.8386897
```

- We see that the two estimates, $\hat{\mu}_1^* = -2.72$ and $\hat{\mu}_2^* = -2.71$, are close. However, $\text{SE}_1 = 0.68$ is an underestimate of error, compared to the better estimate $\text{SE}_2 = 0.84$. The naive standard error needs to be inflated by $100(\text{SE}_2/\text{SE}_1 - 1) = 24.2$ percent.

- Exactly how the ARMA(1,1) model is fitted and the standard errors computed will be covered later.

- We should do **diagnostic analysis**. The first thing to do is to look at the residuals. For an ARMA model, the residual r_n^* at time t_n is defined to be the difference between the data, y_n^*, and a one-step ahead prediction of y_n^* based on $y_{1:n-1}^*$, written as y_n^{P*}. From the ARMA(1,1) definition,

  $$Y_n = \mu + \alpha(Y_{n-1} - \mu) + \epsilon_n + \beta\epsilon_{n-1},$$

  a basic one-step-ahead predicted value corresponding to parameter estimates $\hat{\mu}^*$ and $\hat{\alpha}^*$ could be

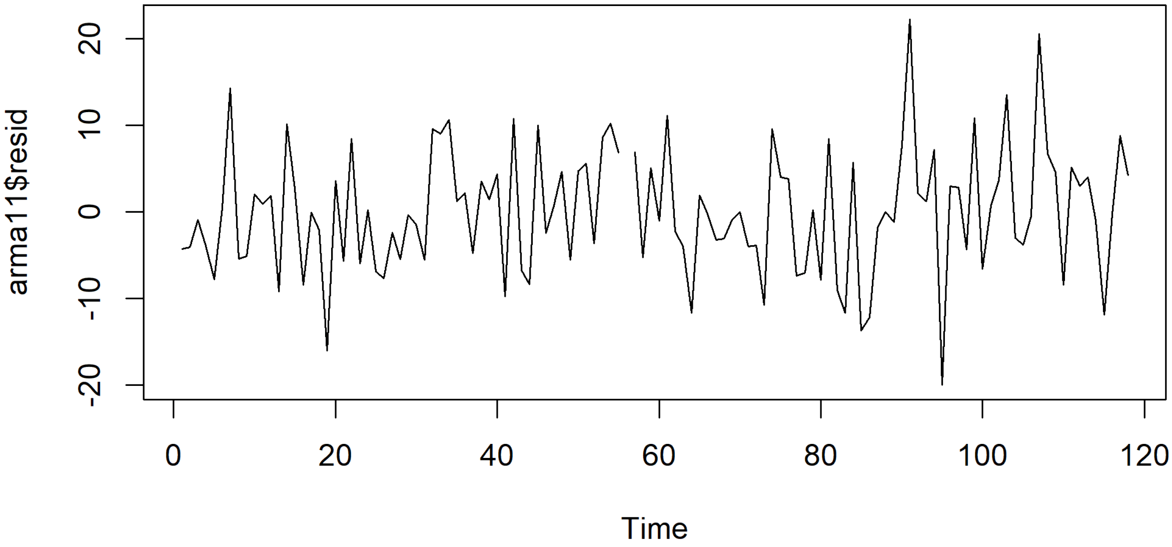
  $$y_n^{P*} = \hat{\mu}^* + \hat{\alpha}^*(y_{n-1}^* - \hat{\mu}^*).$$

  A so-called residual time series, $\{r_n^*\}$, is then given by

  $$r_n^* = y_n^* - y_n^{P*}.$$

  In fact, R does something slightly more sophisticated.

```r
plot(arma11$resid)
```

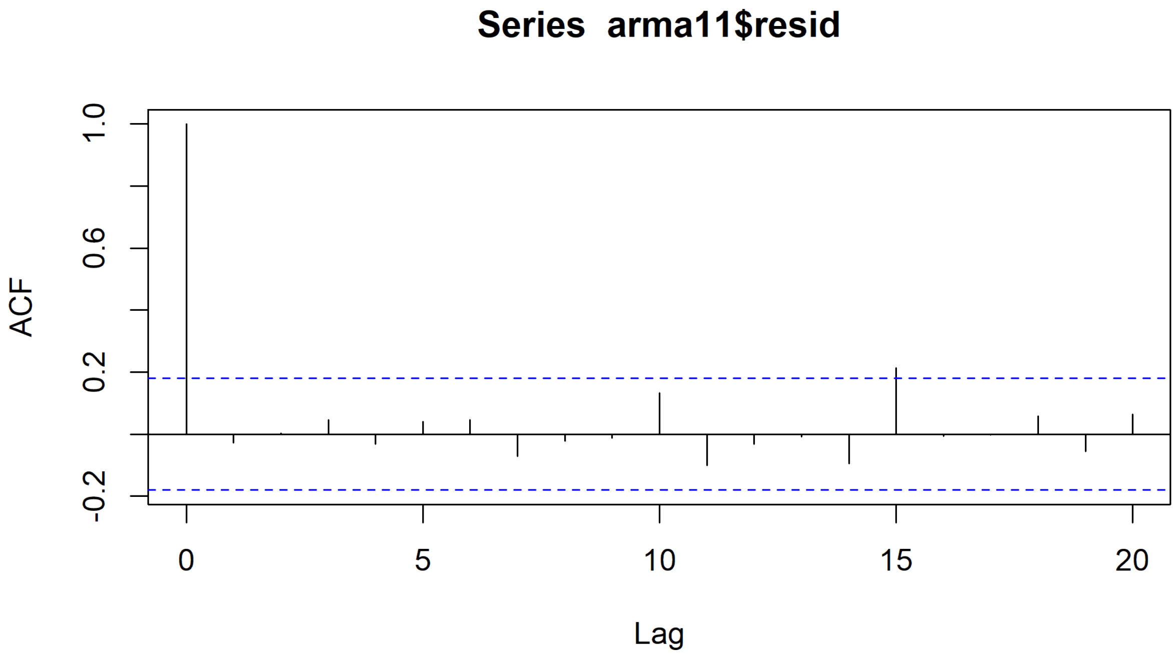


- We see that there seems to be some slow variation in the residuals, over a decadal time scale. However, the residuals are close to uncorrelated, as we can check by plotting their pairwise sample correlations at a range of lags. This is called the sample autocorrelation function, or sample ACF, which can be computed at each lag h for the residual time series $\{r_n^*\}$ as

  $$\mathrm{ACF}^*(h) = \frac{\frac{1}{N}\sum_{n=1}^{N-h} r_n^* \, r_{n+h}^*}{\frac{1}{N}\sum_{n=1}^{N}(r_n^*)^2}.$$

  We will discuss the sample ACF at greater length later.

```r
acf(arma11$resid,na.action=na.pass)
```



- This shows not much sign of autocorrelation. In other words, fitting ARMA models is unlikely to be a good way to describe the slow variation present in the residuals of the ARMA(1,1) model.
- There is probably some room for improvement over SE_2, which might lead to a somewhat larger standard error estimate.
- Although this is just a toy example, the issue of inadequate models giving poor statistical uncertainty estimates is a major concern whenever working with time series data.
- Usually, omitted dependency in the model will give overconfident (too small) standard errors.
  - This leads to scientific reproducibility problems, where chance variation is too often assigned statistical significance.
  - It can also lead to improper pricing of risk in financial markets, a factor in the US financial crisis of 2007-2008.

1.3 Models dynamic systems: State-space models

- Scientists and engineers often have equations in mind to describe a system they're interested in.
- Often, we have a model for how the state of a stochastic dynamic system evolves through time, and another model for how imperfect measurements on this system gives rise to a time series of observations.
- This is called a **state-space model**.
- The **state** models the quantities that we think determine how the system changes with time. However, these idealized state variables are not usually directly and perfectly measurable.
- Statistical analysis of time series data on a system should be able to

1. Help scientists choose between rival hypotheses.
2. Estimate unknown parameters in the model equations.

- We will look at examples from a wide range of scientific applications. The dynamic model may be linear or nonlinear, Gaussian or non-Gaussian. Here is an example from finance.

1.3.1 Fitting a model for volatility of a stock market index

- Let $\{y_n^*, n = 1, \ldots, N\}$ be the daily returns on a stock market index, such as the S&P 500.
- Since the stock market is notoriously unpredictable, it is often unproductive to predict the mean of the returns and instead there is much emphasis on predicting the variability of the returns, known as the **volatility**.
- Volatility is critical to assessing the risk of financial investments.
- Financial mathematicians have postulated the following model. We will not work on understanding it right now. The relevant point here is that investigators often find it useful to write down models for how a dynamic system progresses through time, and this gives rise to the time series analysis goals of estimating unknown parameters and assessing how successfully the fitted model describes the data.

$$\begin{aligned}
Y_n &= \exp\left\{\frac{H_n}{2}\right\}\epsilon_n, \\
H_n &= \mu_h(1-\phi) + \phi H_{n-1} \\
&\qquad + Y_{n-1}\sigma_\eta\sqrt{1-\phi^2}\tanh(G_{n-1}+\nu_n)\exp\left\{\frac{-H_{n-1}}{2}\right\} + \omega_n, \\
G_n &= G_{n-1} + \nu_n,
\end{aligned}$$

- $\{\epsilon_n\}$ is an iid $N(0,1)$ sequence, $\{\nu_n\}$ is an iid $N(0,\sigma_\nu^2)$ sequence, and $\{\omega_n\}$ is an iid $N(0,\sigma_\omega^2)$ sequence.
- H_n represents the volatility at time t_n. Volatility is unobserved; we only get to observe the return, Y_n.
- G_n is a slowly-varying process that regulates H_n.
- H_n has auto-regressive behavior and dependence on the previous return, Y_{n-1}, as well as being driven by G_n.

This is an example of a mechanistic model, where scientific or engineering considerations lead to a model of interest. Now there is data and a model of interest, it is time to recruit a statistician! (Statisticians can have roles to play in data collection and model development, as well.)

1.3.2 Relevant questions to be addressed using methodology covered later in the course

1. How can we get good estimates of the parameters, μ_h, ϕ, σ_ν, σ_ω, together with their uncertainties?
2. Does this model fit better than alternative models? So far as it does, what have we learned?

1.3.3 Outline of a solution

- Likelihood-based inference for this partially observed stochastic dynamic system is possible, and enables addressing these questions (Bretó 2014).
- Carrying out such an analysis is facilitated by recent advances in algorithms (Ionides et al. 2015).
- The R package system and R markdown make state-of-the-art statistical analysis reproducible and extendable by Masters level statisticians! For example, with a modest amount of effort, you could run the code given in the online tutorial reproducing part of Bretó (2014). We will look more at these data and models later.

1.4 Internet repositories for collaboration and open-source research: git and github

- Git is currently the dominant tool for managing, developing and sharing code within the computational sciences and industry. Github is the largest git-based internet repository, but others (such as bitbucket) also use git, and it can be useful to use git to build a local repository on your own computer.
- This course will benefit directly from using git, since it provides a convenient way to manage the code, data, notes and other files involved in the course.
- Also, you will be practicing a skill that will likely be useful for future work.
- Our immediate goals are
    i. Learn some ways to think about what a git repository is and how it works.
    ii. Go through the process of downloading a github repository, editing it, and uploading the changes.
- This introduction uses material from Karl Broman's practical and minimal git/github tutorial (kbroman.org/github_tutorial). A deeper, more technical tutorial is www.atlassian.com/git/tutorials.
- This course will require only basic familiarity with git. Indeed, all the materials will be on the website so it is not essential that you use git at all. However, keeping a local copy of the course git project is a good way to maintain up-to-date copies of all the files. Also, you may find additional features of git to be useful, such as making pull requests with corrections or improvements to the notes!

1.4.1 Getting started with git and github

1. Get an account on github.
2. If you are on a Mac or Linux machine, git will likely be installed already. Otherwise, you can download and install it from git-scm.com/downloads.
3. Set up your local git installation with your user name and email. Open a terminal (or a git BASH window for Windows) and type:

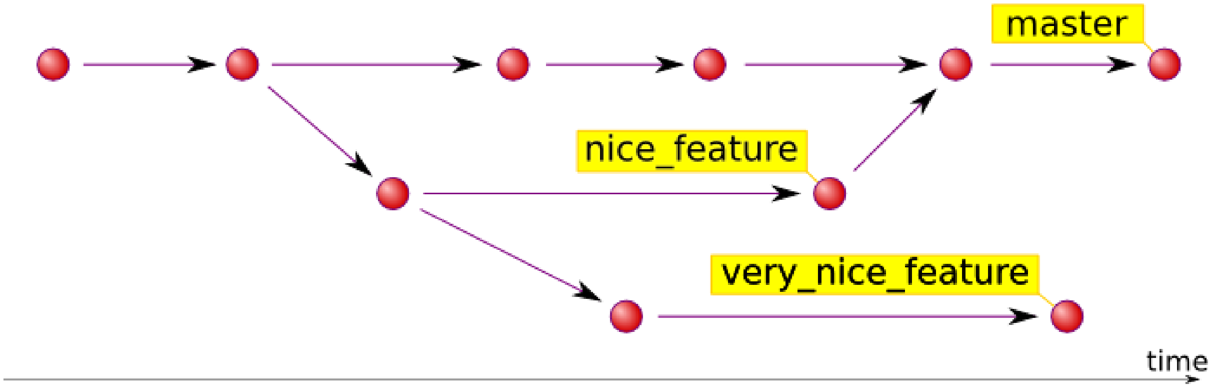
```
$ git config --global user.name "Your name here"
$ git config --global user.email "your_email@example.com"
```

(DonÃ¢Â€Â™t type the $; that just indicates that youÃ¢Â€Â™re doing this at the command line.)

4. Optional but recommended: set up secure password-less SSH communication to github, following the github instructions. If you run into difficulties, it may help to look at Roger Peng's SSH help page.

1.4.2 Basic git concepts

- **repository**. A representation of the current state of a collection of files, and its entire history of modifications.
- **commit**. A commit is a change to one or many of the files in repository. The repository therefore consists of a directed graph of all previous commits.
- **branch**. Multiple versions of the collection of files can exist simultaneously in the repository. These versions are called branches. Branches may represent new functionality, or a bug fix, or different versions of the code with slightly different goals.
  - Branches have names. A special name called **master** is reserved for the main development branch.
  - Branches can be **created**, **deleted** or **merged**.
  - Each new commit is assigned to a branch.
- We now have the pieces in place to visualize the **graph** of a git repository. [Picture credit: hades.github.io]



- Take some time to identify the commits, branching events, and merging events on the graph.
- Note that branch names actually are names for the most recent commit on that branch, known as the **head** of the branch.

1.4.3 An elementary task: cloning a remote repository

- In a suitable directory, type

```
git clone git@github.com:ionides/531w18
```

- You now have a local copy of the Stats 531 class materials. Depending on your setup, it may work better to use a different approach:

```
git clone https://github.com/ionides/531w18
```

- The local repository remembers the address of the remote repository it was cloned from.
- You can pull any changes from the remote repository to your local repository using **git pull**.

```
$ git pull
Already up-to-date.
```

1.4.4 A workflow to contribute to the 531w18 github repository: Forking a project and making a pull request

- We will follow a standard workflow for proposing a change to someone else's github repository.
- **Forking** is making your own github copy of a repository.
- A **pull request** is a way to ask the owner of the repository to pull your changes back into their version.

The following steps guide you through a test example.

1. Go to http://github.com/ionides/531w18
2. Click `fork` at the top right-hand corner, and follow instructions to add a forked copy to your own github account. It should now show up in your account as `my_username/531w18`.
3. Clone a local copy of the forked repository to your machine, e.g.,

```
git clone git@github.com:my_username/531w18
```

Aternatively, you can type

```
git clone https://github.com/my_username/531w18
```

4. Move to the `531w18` directory and edit the file `hw0_signup.html` to add your own name. You should use an ascii text editor, such as Emacs. Do not use Microsoft Word or any other word processing editor.
5. It can be helpful to type

```
git status
```

regularly to check on the current state of the repository.

5. Commit this change to your local version of the forked `531w18`,

```
git add hw0_signup.html
git commit -m "sign up for my_name"
```

and see how the `git status` has changed. Another useful command for checking on the recent action in the repository is

```
git log
```

Type `q` to exit the listing of the log.

6. Push this change to the forked `531w18` on github:

```
git push
```

7. On the github web site for the `my_username/531w18` fork, click `New pull request` and follow instructions. When you have successfully placed your pull request, the owner of the forked repository (in this case, ionides@umich.edu) will be notified. I will then pull the modifications from your fork into `ionides/531w18`.

Optional: upstream

If you have forked and cloned the repository and would like to keep your forked repository up to date, here is a common way of doing so:

- Go to your local repository via the terminal.
- Type `git remote add upstream https://github.com/ionides/531w18`
  - This allows you to have 2 remote branches, 'origin' which points to the forked repository that you have, and 'upstream' which is the original repository. Recall that remote branches are references (pointers) to the state of branches in remote repositories. Thus, you can pull via 'upstream' to update the local repository on your machine, then push from your updated local repository to the one on github.
- Type `git pull upstream master`. This pulls the changes in the upstream repository (the original repository) into the master branch on your local computer
  - If you have already made changes to your local repository, there *might* be conflicts when you pull because the upsteam repository doesn't have those changes. You'll have to manage these manually if this occurs.
- Type `git push origin master`. This, as mentioned above, pushes changes from your local repository, which now contains updates from the original repository, to your repository on github. Now, both your local repository and your github repository should be updated.

References

Bretó, C. 2014. On idiosyncratic stochasticity of financial leverage effects. Statistics & Probability Letters 91:20–26.

Ionides, E. L., D. Nguyen, Y. Atchadé, S. Stoev, and A. A. King. 2015. Inference for dynamic and latent variable models via iterated, perturbed Bayes maps. Proceedings of the National Academy of Sciences of the U.S.A. 112:719–724.