ETH zürich

KOF

Methods of Macroeconomic Forecasting

A Forecasting Baseline - Lab 1

KOF ETH Zurich

October 2, 2025, Zurich

Lab 1 Overview

In Lab 1, we'll look at:

1. An introduction to Git, GitHub and R.
2. A forecasting baseline.

Git, Github and R

Authenticating on GitHub

- Generate an SSH Key pair:

```
# bash
ssh-keygen -t ed25519 -C "your_email@example.com"
```

- Start SSH agent and add key

```
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_ed25519
```

- Copy and save public key to GitHub

```
cat ~/.ssh/id_ed25519.pub
```

  Go to GitHub -> Settings -> SSH and GPG keys

- Test connection

```
ssh -T git@github.com
```

Git and GitHub

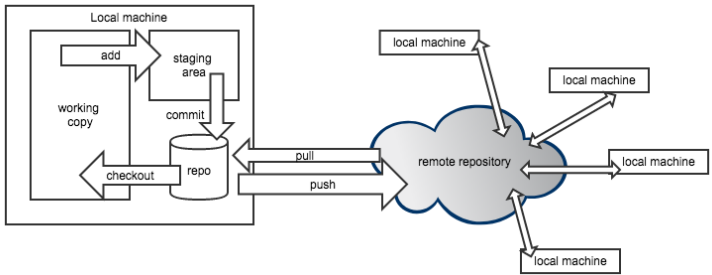


Figure: Schematic illustration of an example git workflow including remote repositories (from Bannert, 2024, Fig. 5.3)

Bannert, M. (2024). Research Software Engineering: A Guide to the Open Source Ecosystem. Chapman & Hall/CRC.

Git commands - local

```
# initialize git repo in dir
git init

# shows status
git status

# adds file to tracked files
git add filename.py

# creates a new version/commit out of all staged files
git commit -m "meaningful msg"
```

- .gitignore files that you don’t want to share

Git commands - remote

```
# copy a remote repo to local machine
git clone

# update local repo with remote changes
git pull

# upload local commits to remote repo
git push

# download objects/refs, don't merge
git fetch
```

Setting up R

- Installing R
  - Download and install binary: CRAN
  - Or install using a package manager:
    - works on all platforms: R Installation Manager **rig**
    - macOS and linux: The Multiple Runtime Version Manager: **asdf**
    - macOS and linux **Homebrew**

You are free to install and manage R in your preferred way.

Reproducibility with R

```
# R
install.packages("MASS")
```

Reproducibility with R

```
# R
install.packages("MASS")
```

- installs packages to library path

```
.libPaths()
```

Reproducibility with R

```
# R
install.packages("MASS")
```

- installs packages to library path

```
.libPaths()
```

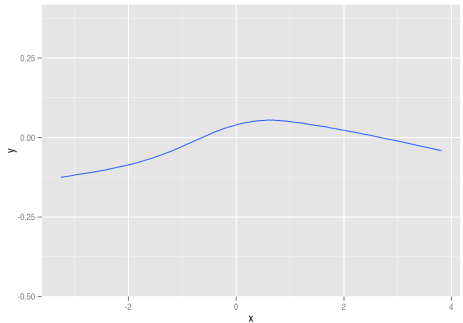
- when working with others we want to work on same package versions to avoid dependency issues
  → we will be using **renv** for package management

Versions matter

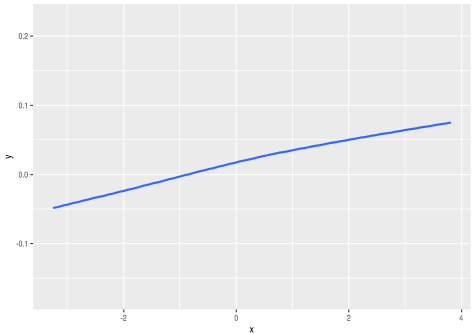
```
library(ggplot2)
set.seed(1) # fix random seed
df <- data.frame(x = rnorm(2000), y = rnorm(2000))
ggplot(df, aes(x, y)) + stat_smooth()
```

Versions matter

```
library(ggplot2)
set.seed(1) # fix random seed
df <- data.frame(x = rnorm(2000), y = rnorm(2000))
ggplot(df, aes(x, y)) + stat_smooth()
```



(a) ggplot2@0.9.3



(b) ggplot2@4.0.0

renv commands

```
# R
renv::init() # setup renv
```

renv commands

```
# R
renv::init() # setup renv
```

```
renv::status() # see if the packages are synchronized
```

renv commands

```
# R
renv::init() # setup renv
```

```
renv::status() # see if the packages are synchronized
```

```
renv::install("ggplot@0.9.3") # install specific package versions
```

renv commands

```
# R
renv::init() # setup renv
```

```
renv::status() # see if the packages are synchronized
```

```
renv::install("ggplot@0.9.3") # install specific package versions
```

```
renv::snapshot() # snapshot currently installed packages to lock
    file
```

renv commands

```
# R
renv::init() # setup renv
```

```
renv::status() # see if the packages are synchronized
```

```
renv::install("ggplot@0.9.3") # install specific package versions
```

```
renv::snapshot() # snapshot currently installed packages to lock
    file
```

```
renv::restore() # restore packages to versions in lock file
```

- Additional benefit of using *renv* is package caching

renv.lock

```
"R": {
  "Version": "4.5.1",
  "Repositories": [
    {
      "Name": "CRAN",
      "URL": "https://packagemanager.posit.co/cran/latest"
    }
  ]
},
"Packages": {
  "R6": {
    "Package": "R6",
    "Version": "2.6.1",
    "Source": "Repository",
    "Title": "Encapsulated Classes with Reference Semantics",
```

Docker

- A tool to package software and dependencies into containers
- Ensures code runs the same everywhere

Docker

- A tool to package software and dependencies into containers
- Ensures code runs the same everywhere

Why use it?

- Reproducibility across machines
- Easy sharing of environments
- Isolation: no conflicts between projects

Lab 1 Overview

In Lab 1, we'll look at:

1. An introduction to Git, GitHub and R. ✓
2. A forecasting baseline.

Forecasting GDP with AR models
A Forecasting Baseline

Motivation I

- GDP forecasts are often a key part of assessing the developments of an economy.
- Producing reliable forecasts is thus important to a variety of economic agents.
- What are the best methods to produce accurate forecasts?

Motivation II

Two key arguments for an AR baseline:

1. We need a model to compare if our (new) approach improves forecasting accuracy (via evaluation metrics).
2. Simple models often perform really well and allow easy interpretability (which is really important for economic analysis).

Methodology

AR(1) process:

$$y_t = c + \phi y_{t-1} + \varepsilon_t, \text{ with } \varepsilon_t \sim N(0, \sigma_\varepsilon^2).$$

- Intuition: Today's value depends on yesterday's value plus a shock.
- A simple model to capture persistence, which is determined by ϕ.
- Both frequentist and Bayesian estimation.

Data

Swiss GDP:

- Data obtained from SECO (State Secretariat of Economic Affairs): *Quarterly aggregates of Gross Domestic Product, production approach*.[1]
- Data is seasonally and sports event adjusted.
- Quarterly frequency.
- Growth rates are computed as quarter-on-quarter percentage changes.
- Data is available from 1980 Q1 - 2025 Q2.

[1] See: GDP data.

Forecasting Approach

Forecasting Swiss quarterly GDP growth:

- Forecast horizon: 1-quarter ahead forecasts.
- Rolling window estimation, based on an in-sample period of 80 quarters (i.e. 20 years).

Results and Interpretation

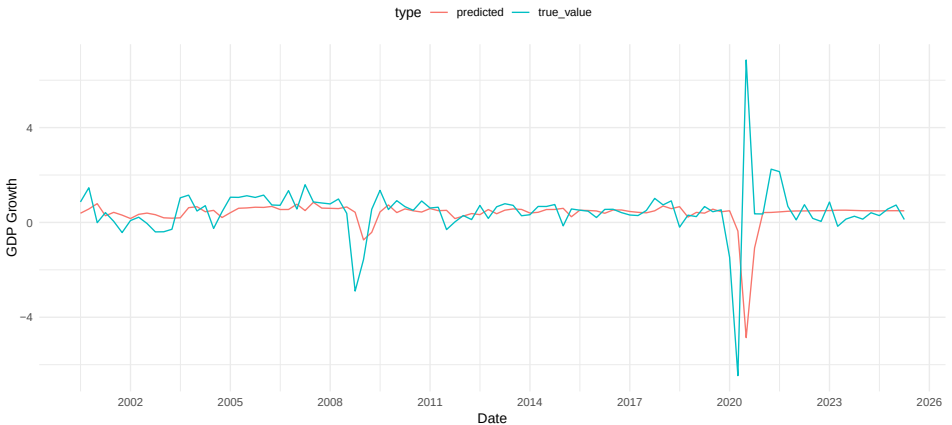


Figure: Rolling window one-step ahead forecast of gdp growth via AR(1) model.

Discussion

- **Advantages:** Simple and easy to interpret.
- **Limitation:** Does not take any other variables into account.

Your Presentation

Some notes/tips:

- Know your data!
- Work with visualisations.
- Provide intuition for what you are doing (both in terms of the methodology and the results).
- Be able to explain the formulas which you show.
- Think about the audience you are presenting to (technical and policy audience).
- You can turn to the literature to get an idea of how common issues are usually addressed.
- You do not need to jump to the code in your presentation, but it needs to be publicly available by **19 November 2025** on GitHub.