Reproductibility of the CV is dead, long live the CV!

Version 1 (march 2023)

2023-03-02

Abstract

THE purpose of this document is to allow for the reproducibility of the figure and the statistical analyses presented in the article (TODO: ref). It was written in `RMarkdown` (Xie, Allaire, and Grolemund 2018; Xie, Dervieux, and Riederer 2020; Allaire et al. 2020) and compiled with R package `knitr` (Xie 2015, 2014, 2020a). The R package `tinytex` (Xie 2019, 2020b) was used to install the TinyTEXdistribution. The LATEX packages and macros used are given in `preamble.tex` printed in section 9.2 page 23 .

Contents

1 Statistics definitions and implementations

1.1 Time series notation

Let $\boldsymbol{x}$ be a time-serie of n elements. In masting studies the elements are from $\mathbb{R}_+$, that is, we are dealing with *non negative* time-series:

$\forall i \in \{1, \ldots, n\} : x_i \geq 0$

Note that all statistics, S, used here share the scale-invariance property, which means basically that we want to see the same variability in $(1, 2, 3)$ than in $(100, 200, 300)$:

$$\forall \lambda \in \mathbb{R}_+^*, \forall \boldsymbol{x} \in (\mathbb{R}_+)^n : \mathrm{S}(\lambda \boldsymbol{x}) = \mathrm{S}(\boldsymbol{x})$$

1.2 Numerical example

Table 1332. Foreign or Foreign-Born Population, Labor Force, and Net Migration in Selected OECD Countries: 2000 and 2007

[31,108 represents 31,108,000. In Australia and the United States, the data refer to people present in the country who are foreign born. In the European countries and Japan, they generally refer to foreigners and represent the nationalities of residents. Minus sign (–) indicates net loss]

Country	Foreign population [1]				Foreign labor force [2]				Average net migration 1990–2007 [3]
	Number (1,000)		Percent of total population		Number (1,000)		Percent of total population		
	2000	2007	2000	2007	2000	2007	2000	2007	(per 1,000 population)
United States	**31,108**	**41,100**	**11.0**	**13.6**	**18,029**	**24,778**	**12.9**	**16.3**	**4.0**
Australia	4,412	5,254	23.0	25.0	2,373	2,827	24.7	25.8	5.7
Austria	702	840	8.7	10.1	346	452	10.6	13.1	3.9
Belgium	862	971	8.4	9.1	388	449	8.6	9.5	3.2
Denmark	259	299	4.8	5.5	97	127	3.4	4.4	2.2
France	(NA)	(NA)	(NA)	(NA)	1,578	1,486	6.0	5.4	1.3
Germany	7,297	6,745	8.9	8.2	3,546	3,874	8.8	9.4	3.0
Italy [4]	1,380	3,433	2.4	5.8	838	1,638	3.9	6.6	3.8
Japan [5]	1,686	2,151	1.3	1.7	155	194	0.2	0.3	–0.1
Luxembourg	165	206	37.3	43.2	153	222	58.0	66.6	9.8
Netherlands	668	688	4.2	4.2	300	314	3.9	3.6	1.8
Spain [6]	1,371	5,221	3.4	11.6	455	1,981	2.5	9.0	7.0
Sweden	477	525	5.4	5.7	222	(NA)	5.0	(NA)	3.0
Switzerland [7]	1,384	1,571	19.3	20.8	717	876	20.1	21.3	4.3
United Kingdom [8]	2,342	3,824	4.0	6.5	1,107	2,035	4.0	7.2	0.7

Figure 1: The table used to build the numerical example.

In his paper section 3.2 page 408 Kvålseth gives the following numerical application: "[f]or example, the U.S. Census Bureau [23, Table 1332] lists the foreign or foreign-born populations in 14 different countries for which one can compute the V_2 as $V_2 = 0.89$ or 89%." This table is reproduced here in figure 1 page 3 . The data are for the census of year 2007[1]:

```
xK <- c(41100, 5254, 840, 971, 299, NA, 6745,
        3433, 2151, 206, 688, 5221, 525, 1571, 3824)
```

This is not a time series *per se* but it does not matter for numerical applications since all statistics used here are not dependent on the order of values in $\boldsymbol{x}$.

[1] Tarald O. Kvålseth, personnal communication

It contains a missing value and this is intresting to check that those are handled correctly.

1.3 Pearson's coeffcient of variation $^{\mathrm{P}}$CV

Pearson's coefficient of variation (Pearson 1896), $^{\mathrm{P}}$CV, is the ratio of the sample standard deviation, s_x, over the sample mean, $\bar{x}$:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad s_x^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad {}^{\mathrm{P}}\mathrm{CV}(\boldsymbol{x}) = \frac{s_x}{\bar{x}}$$

Note that $^{\mathrm{P}}$CV is not defined when the mean is zero. Because of the non-negativness of masting time-series this happens for us only with the null vector. This is a serious issue here because of the zero-inflatedness of time-series in masting studies: during bootstrap re-sampling it's not unlikely to obtain the null vector. Moreover, there are *documented* time-series corresponding to the null vector. Out of the 1433 quantitative series with at least 12 values in MASTREE+ there are 3 null vectors (*viz.* `6196-001-01-EUCGRA` for *Eucalyptus grandis* shared by Urs Kalbitzer and Colin Chapman in (Hacket-Pain et al. 2022), `6193-001-01-CARSPI` for *Carissa spinarum* and `6193-001-01-DORSTI` for *Doratoxylon stipulatum* shared by Amy. E. Dunham *et al.*[2]). By convention $^{\mathrm{P}}\mathrm{CV}(\mathbf{0}) = 0$, its value is forced to zero when the mean is zero:

$${}^{\mathrm{P}}\mathrm{CV}(\boldsymbol{x}) = \begin{cases} 0 & \text{if} \quad \bar{x} = 0 \\ \frac{s_x}{\bar{x}} & \text{if} \quad \bar{x} \neq 0 \end{cases}$$

To compute the coefficient of variation $^{\mathrm{P}}$CV we need the sample standard deviation, s_x. The function `sd()` from R compute $\hat{\sigma}$, the unbiased estimate of the population standard deviation:

$$\hat{\sigma} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} = s_x\sqrt{\frac{n}{n-1}}$$

Since there is no argument in function `sd()` to switch to the sample standard deviation, we need to define our own `sdn()` function for this purpose. We want this function to handle correctly missing values and usable directly to compute the statistic or indirectly as an argument to the `boot()` function from the eponymous R base package (Canty and Ripley 2021; Davison 1997) to compute confidence intervals.

```
sdn <- function(x, i, ...){
  n <- sum(!is.na(x)) # number of non missing values
  return(sqrt((n - 1)/n)*sd(x[i], ...))
}
```

[2]Dunham, Amy E. et al. (2018), Data from: Fruiting phenology is linked to rainfall variability in a tropical rain forest, Dryad, Dataset, https://doi.org/10.5061/dryad.61m318c

A function given as an argument to `boot()` must have as the first argument the vector of observation `x` and as the second one the vector `i` of the rank of re-sampled values. Note that the function `sdn()` will work even if `i` is missing because `x[ ]` returns `x`. We check first that missing values are correctly handled via the dot-dot-dot argument to pass the `na.rm` argument to `sd()` function when necessary:

```
sdn(xK)
```

```
[1] NA
```

```
sdn(xK, na.rm = TRUE)
```

```
[1] 10165.59
```

We illustrate then how to use the `boot()` function to compute the confidence interval. The pargument `R` is the number of bootstap sample generated. The `type = "bca"` argument correspond to the confidence interval type recommended (Efron 1987).

```
library(boot)
boot.out <- boot(xK, sdn, R = 9999, na.rm = TRUE)
boot.ci(boot.out, type = "bca")$bca[4:5]
```

```
[1]  1881.424 18898.273
```

To define the function `PCV()` we use the standard `all.equal()` function that allows to test if the mean is nearly equal (taking into account the numerical accuracy available in the platform in use) to zero.

```
PCV <- function(x, i, ...){
  barx <- mean(x[i], ...)
  if(isTRUE(all.equal(barx, 0))) return(0)
  return(sdn(x[i], ...)/barx)
}
PCV(xK, na.rm = TRUE)
```

```
[1] 1.95417
```

```
# Sanity checks
xnull <- rep(0.0, 10)
stopifnot(all.equal(PCV(xnull), 0.0))
stopifnot(is.na(PCV(xK)))
stopifnot(all.equal(PCV(xK), PCV(pi*xK)))
```

We can now compute the confidence intervals.

```r
set.seed(1)
boot.out <- boot(xK, PCV, R = 9999,
                 parallel = "multicore", ncpus = 10, na.rm = TRUE)
boot.ci(boot.out, type = "bca")$bca[4:5]
```

```
[1] 1.521883 2.898294
```

1.4 Kvålseth's coefficient of variation $^{\mathrm{K}}$CV

THE statistic $^{\mathrm{K}}$CV (noted V_2 in the original paper, we do not use this notation here because in masting studies we need to save the right index spot to note individual or populationnal statistics such as $^{\mathrm{K}}\mathrm{CV_i}$ or $^{\mathrm{K}}\mathrm{CV_p}$) was introduced relatively recently (Kvålseth 2017) as an alternative to PEARSON's coefficient of variation $^{\mathrm{P}}$CV defined in section 1.3 page 4 . The sample standard deviation is not divided by the mean but by the square root of the mean of squared values. It is defined for all $\boldsymbol{x} \in \mathbb{R}^n$ except for the null vector $\mathbf{0}$:

$$\forall \boldsymbol{x} \neq \mathbf{0} : {}^{\mathrm{K}}\mathrm{CV}(\boldsymbol{x}) = \frac{s_x}{\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}}$$

As previously, we will force the value to zero for the null vector[3]:

$${}^{\mathrm{K}}\mathrm{CV}(\mathbf{0}) = 0$$

The $^{\mathrm{K}}$CV statistic can be rewritten (Kvålseth 2017) to show that it is bounded between 0 and 1:

$$\forall \boldsymbol{x} \neq \mathbf{0} : {}^{\mathrm{K}}\mathrm{CV}(\boldsymbol{x}) = \sqrt{\frac{s_x^2}{s_x^2 + \bar{x}^2}}$$

It can also be seen as a variance-stabilization transform of $^{\mathrm{P}}$CV:

$${}^{\mathrm{K}}\mathrm{CV} = \sqrt{\frac{{}^{\mathrm{P}}\mathrm{CV}^2}{1 + {}^{\mathrm{P}}\mathrm{CV}^2}}$$

WE use this last expression to define the `KCV()` function and check that the null vector and missing values are handled correctly and the scale invariance property.

[3] "your proposal for the null vector seems reasonable, although such a situation would seem to be rather unusual", Tarald O. KVÅLSETH, personnal communication

```
KCV <- function(x, i, ...){
  PCV2 <- PCV(x, i, ...)^2
  return(sqrt(PCV2/(1 + PCV2)))
}
KCV(xK, na.rm = TRUE) # 0.89 in Kvalseth's paper
```

```
[1] 0.8902127
```

```
# Sanity checks
stopifnot(all.equal(KCV(xnull), 0.0))
stopifnot(is.na(KCV(xK)))
stopifnot(all.equal(KCV(xK), KCV(pi*xK)))
```

We can now compute the confidence interval.

```
set.seed(1)
boot.out <- boot(xK, KCV, R = 9999,
                 parallel = "multicore", ncpus = 10, na.rm = TRUE)
boot.ci(boot.out, type = "bca")$bca[4:5]
```

```
[1] 0.8382567 0.9453138
```

1.5 Heath's proportional variability PV

This statistic, PV, was introduced first as the acronym of *Population Variability* (Heath 2006) and then as the acronym of *Proportional Variability* (Heath and Borowski 2013). The latter is less confusing since PV could be used to quantify variability both at the individual and populational level. PV is based on a metric *sensu stricto*, $d(x_i, x_j)$, defining the distance between two values x_i and x_j in the time-serie $\boldsymbol{x}$:

$$d(x_i, x_j) = \begin{cases} 0 & \text{if} \quad x_i = x_j \\ \frac{|x_i - x_j|}{\max(x_i, x_j)} & \text{if} \quad x_i \neq x_j \end{cases}$$

$$\iff \quad d(x_i, x_j) = \begin{cases} 0 & \text{if} \quad x_i = x_j \\ \frac{\max(x_i, x_j) - \min(x_i, x_j)}{\max(x_i, x_j)} & \text{if} \quad x_i \neq x_j \end{cases}$$

$$\iff \quad d(x_i, x_j) = \begin{cases} 0 & \text{if} \quad x_i = x_j \\ 1 - \frac{\min(x_i, x_j)}{\max(x_i, x_j)} & \text{if} \quad x_i \neq x_j \end{cases}$$

With the last expression it is obvious that d values are in the range $[0, 1]$. Note that $d(0, 0)$ is defined as a special case of $x_i = x_j$. The statistic PV is the arithmetic mean of all $\frac{1}{2}n(n-1)$ distinct possible pairs (x_i, x_j):

$$\mathrm{PV}(\boldsymbol{x}) = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d(x_i, x_j)$$

Define the function `d()` to compute the distance used by PV:

```
d <- function(x, y){
  if(isTRUE(all.equal(x, y,
                      check.names = FALSE,
                      check.attributes = FALSE))){
    return(0)
  } else {
    return(abs(x - y)/max(x, y))
  }
}
```

We define then the function `PV()`.

```
PV <- function(x, i, ...){
  n <- length(x)
  xx <- x[i] # bootstrap sample sample of x if i is missing
  dists <- numeric(n*(n-1)/2) # all pairwise distances
  ii <- 1
  for(k in seq_len(n - 1)){
    for(l in (k + 1):n){
      dists[ii] <- d(xx[k], xx[l])
      ii <- ii + 1
    }
  }
  return(mean(dists, ...))
}
PV(xK, na.rm = TRUE)
```

```
[1] 0.6883436
```

```
# Sanity checks
stopifnot(all.equal(PV(xnull), 0.0))
stopifnot(is.na(PV(xK)))
stopifnot(all.equal(PV(xK), PV(pi*xK)))
```

We can now compute the confidence interval.

```
set.seed(1)
boot.out <- boot(xK, PV, R = 9999,
                 parallel = "multicore", ncpus = 10, na.rm = TRUE)
save(boot.out, file = "batch/bootPV.Rda")
```

```
load("batch/bootPV.Rda")
boot.ci(boot.out, type = "bca")$bca[4:5]
```

```
[1] 0.6415592 0.7551986
```

2 Table 1

Dataset number	Ten-year dataset										PV	P_{CV}	K_{CV}
1	5	5	5	5	5	5	5	5	5	1000	0.20	2.86	0.94
2	995	995	995	995	995	995	995	995	995	0	0.20	0.33	0.32
3	1	2	3	4	5	6	7	8	9	1000	0.60	2.86	0.94
4	0	0	0	0	0	0	0	0	0	1000	0.20	3.00	0.95

THE four ten-years time series used in table 1 are defined in the following R. They are set in the list object `x` to facilitate statistics computation thereafter.

```
x <- vector("list", 4)
(x[[1]] <- c(rep(5, 9), 1000))
```

```
 [1]    5    5    5    5    5    5    5    5    5 1000
```

```
(x[[2]] <- c(rep(995, 9), 0))
```

```
 [1] 995 995 995 995 995 995 995 995 995   0
```

```
(x[[3]] <- c(1:9, 1000))
```

```
 [1]    1    2    3    4    5    6    7    8    9 1000
```

```
(x[[4]] <- c(rep(0, 9), 1000))
```

```
 [1]    0    0    0    0    0    0    0    0    0 1000
```

```
cbind(PV  = round(sapply(x, PV), 2),
      PCV = round(sapply(x, PCV), 2),
      KCV = round(sapply(x, KCV), 2))
```

```
      PV  PCV  KCV
[1,] 0.2 2.86 0.94
[2,] 0.2 0.33 0.32
[3,] 0.6 2.86 0.94
[4,] 0.2 3.00 0.95
```

3 Figure 1

3.1 Dataset

We use the data from MASTREE+ (Hacket-Pain et al. 2022) and select quantitative time series with at least 12 documented values. The version used here is from the file `MASTREEplus_2022-02-03_V1.RData` from the github repository. A primary key, `ID`, unique for each time-serie, was built by concatenation of `Alpha_Number`, `Site_number`, `Variable_number` and `Species_code`.

```
load("data/mastree.Rda")
ms <- subset(mastree, VarType == "C" & Length >= 12)
length(unique(ms$ID))
```

```
[1] 1433
```

We save all data and functions defined up to now into the file `myws.Rda` in the `batch` folder. This is to reload easilly our workspace during batch computations.

```
save.image(file = "batch/myws.Rda")
```

3.2 Pearson's ^{P}CV

In order to speed-up computations, we use here a parallelization by data approach. The idea is simple: since all computations are independant for each series we split the table of results into segments to be computed by each process. We have then a master process used to dispatch the work:

batch/dispatchPCV.R

```
load("myws.Rda") # load data and functions

nproc <- 10 # Number of process used
R <- 9999 # Size of bootstrap samples
IDs <- unique(ms$ID) ; n <- length(IDs)
# Spliting the work in n segments from iinf to isup
ii <- floor(seq(from = 1, to = n, length.out  = nproc + 1))
iinf <- ii[-length(ii)]
isup <- ii[-1] - 1
isup[length(isup)] <- n # Not miss the last one

genericLines <- readLines("generic.R")

for(i in seq_len(nproc)){
  # Generating R scripts
  fname <- paste0("genericPCV-", i, ".R")
  codeR <- c(paste0("iproc <- ", i),
             paste0("iseq <- ", iinf[i], ":", isup[i]),
             paste0("Stat <- 'PCV'"),
             paste0("R <- ", R),
             genericLines)
  writeLines(codeR, fname)
  # Generation the command line to run the R scripts
  cmd <- paste0("nohup R CMD BATCH ", fname, "&")
  system(cmd)
}
```

To run the script we use the following command line on unix-like systems:

```
unix$ R CMD BATCH dispatchPCV.R
```

This script will run in parallel as many processes than the number required (variable `nproc`), for instance for number 3 the corresponding script is:

batch/genericPCV-3.R

```
iproc <- 3
iseq <- 287:429
Stat <- 'PCV'
R <- 9999
# Generic part to compute stat S
load("myws.Rda")
IDs <- unique(ms$ID) ; n <- length(iseq)
tabres <- as.data.frame(matrix(NA, nrow = n, ncol = 4))
colnames(tabres) <- c("ID", "est", "inf", "sup")
S <- get(Stat)
tabname <- paste0("tab-", iproc, Stat)
set.seed(1) # for reproducibility
alpha <- 0.05
for(i in seq_len(n)){
  the_ID <- IDs[iseq[i]]
  tabres[i, "ID"] <- the_ID
  x <- ms[ms$ID == the_ID, "Value"]
  try.boot <- try(boot::boot(x, S, R = R))
  if(!inherits(try.boot, "try-error")) tabres[i, "est"] <- try.boot$t0
  try.ci <- try(boot::boot.ci(try.boot, type = "bca"))
  if(!inherits(try.ci, "try-error")) tabres[i, c("inf", "sup")] <- try.ci$bca[4:5]
}

assign(tabname, tabres)
save(list = tabname, file = paste0(tabname, ".Rda"))
```

The process number `iproc` allow us to build different names for the table of result for each segment (`tab-1PCV.Rda`, `tab-2PCV.Rda`, `tab-3PCV.Rda`, etc.) to avoid collisions during results saving. The sequence `iseq` gives the rank of series in charge of the process, number 3 compute PCV values for series from 287 to 429. When the computation are done we just have to gather all segments in a single table:

```
fics <- dir(path = "batch",
            pattern = glob2rx("tab-*PCV.Rda"), full.names = TRUE)
for(i in seq_len(length(fics))) load(fics[i])
tabs <- ls(pattern = glob2rx("tab-*PCV"))
eisPCV <- do.call("rbind", lapply(tabs, get))
save(eisPCV, file = "batch/PCV.Rda")
rm(list = tabs) # clean workspace
```

The table of results shows that it was always possible to compute $^{\mathrm{P}}$CV. The confidence interval wasn't obtained for three series already listed in section 1.3 page 4 and corresponding to a null vector:

```
load("batch/PCV.Rda")
eisPCV[is.na(eisPCV$est), "ID"]
```

```
character(0)
```

```
eisPCV[is.na(eisPCV$inf), "ID"]
```

```
[1] "6196-001-01-EUCGRA" "6193-001-01-CARSPI" "6193-001-01-DORSTI"
```

3.3 Kvålseth's ^{K}CV

The following dispatching script was used:

batch/dispatchKCV.R

```
load("myws.Rda") # load data and functions

nproc <- 10 # Number of process used
R <- 9999 # Size of bootstrap samples
IDs <- unique(ms$ID) ; n <- length(IDs)
# Spliting the work in n segments from iinf to isup
ii <- floor(seq(from = 1, to = n, length.out  = nproc + 1))
iinf <- ii[-length(ii)]
isup <- ii[-1] - 1
isup[length(isup)] <- n # Not miss the last one

genericLines <- readLines("generic.R")

for(i in seq_len(nproc)){
  # Generating R script
  fname <- paste0("genericCV-", i, ".R")
  codeR <- c(paste0("iproc <- ", i),
             paste0("iseq <- ", iinf[i], ":", isup[i]),
             paste0("Stat <- 'KCV'"),
             paste0("R <- ", R),
             genericLines)
  writeLines(codeR, fname)
  # Generation the command line to run the R scripts
  cmd <- paste0("nohup R CMD BATCH ", fname, "&")
  system(cmd)
}
```

```
fics <- dir(path = "batch",
            pattern = glob2rx("tab-*KCV.Rda"), full.names = TRUE)
for(i in seq_len(length(fics))) load(fics[i])
tabs <- ls(pattern = glob2rx("tab-*KCV"))
eisKCV <- do.call("rbind", lapply(tabs, get))
save(eisKCV, file = "batch/KCV.Rda")
rm(list = tabs)
```

```
load("batch/KCV.Rda")
eisKCV[is.na(eisKCV$est), "ID"]
```

```
character(0)
```

```
eisKCV[is.na(eisKCV$inf), "ID"]
```

```
[1] "6196-001-01-EUCGRA" "6193-001-01-CARSPI" "6193-001-01-DORSTI"
```

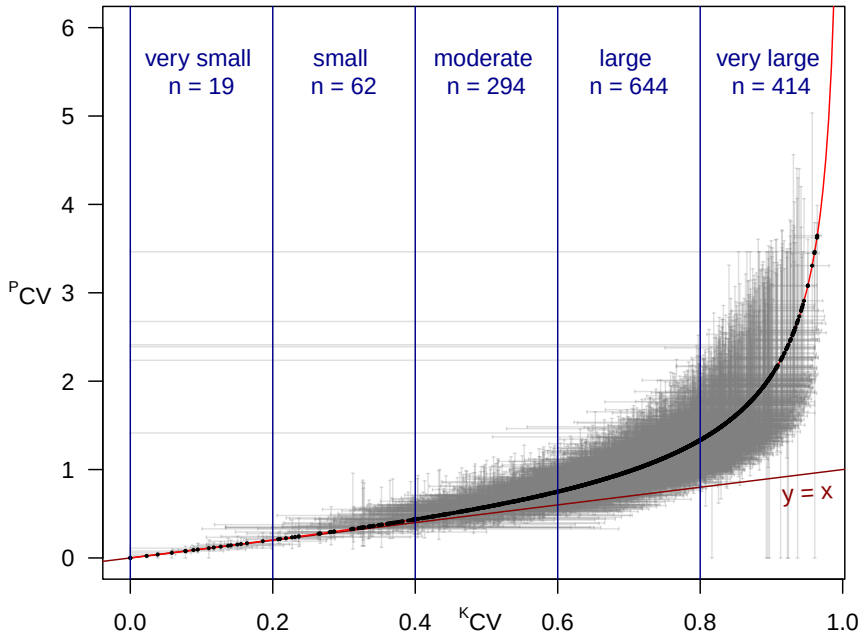
3.4 Figure code

```
load("batch/PCV.Rda") ; load("batch/KCV.Rda")
myarrows <- function(x0, y0, x1, y1,
                     length = 0.01, angle = 90, code = 3,
                     col = rgb(0.5, 0.5, 0.5, 0.25), ...){
  arrows(x0, y0, x1, y1, length, angle, code, col, ...)
}

nser <- nrow(eisPCV)
par(mar = c(2, 5, 0, 1) + 0.1)
plot(eisKCV$est, eisPCV$est, pch = 19, cex = 0, las = 1,
     ylab = "", main = "", xlab = "", ylim = c(0, 6))
text(-0.15, 3, expression(phantom(0)^P*CV), xpd = NA)
title(xlab = expression(phantom(0)^K*CV), line = 1)
myarrows(eisKCV$inf, eisPCV$est, eisKCV$sup, eisPCV$est)
myarrows(eisKCV$est, eisPCV$inf, eisKCV$est, eisPCV$sup)
abline(c(0, 1), col = "darkred")
text(0.95, 0.97, "y = x", pos = 1, col = "darkred", srt = 4)
curve(sqrt(x^2/(1 - x^2)), add = TRUE, col = "red",
      from = 0.0, to = 0.9999, n = 256)
points(eisKCV$est, eisPCV$est, pch = 19, cex = 0.25)

abline(v = seq(0, 0.9, by = 0.2), col = "darkblue")

KvalsethBreaks <- seq(0, 1, by = 0.2)
KvalsethVerb <- c("very small", "small", "moderate",
                  "large", "very large")
nts <- table(cut(eisKCV$est, breaks = KvalsethBreaks,
                 include.lowest = TRUE))
msg <- paste(KvalsethVerb, "\nn =", nts)
text(KvalsethBreaks[-1] - 0.1, 5.5, msg, col = "darkblue")
```



The red line is the curve with equation:

$$ {}^{\mathrm{P}}\mathrm{CV} = \sqrt{\frac{{}^{\mathrm{K}}\mathrm{CV}^2}{1 - {}^{\mathrm{K}}\mathrm{CV}^2}} $$

4 Figure 2

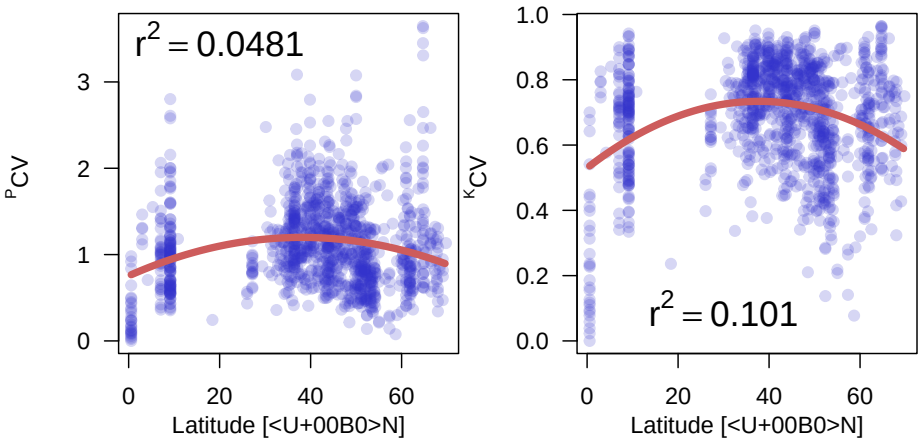
```
msu <- ms[!duplicated(ms$ID), ]
msu <- merge(msu, eisKCV,
                by.x = "ID", by.y = "ID")
msu <- merge(msu, eisPCV,
                by.x = "ID", by.y = "ID")
north <- subset(msu, Latitude > 0)
nrow(north)
```

```
[1] 1138
```

```
myplot <- function(north, stat, xr, yr){
  x <- north$Latitude
  if(stat == "PCV"){
    y <- north$est.y
    ylab <- expression(phantom(0)^P*CV)
  } else {
    y <- north$est.x
```

```
    ylab <- expression(phantom(0)^K*CV)
  }
  par(mar = c(4, 4, 1, 0) + 0.1, pty = "s")
  plot(x, y, main = "",
                ylab = ylab, las = 1, xlab = "",
                pch = 19, col = rgb(0.2, 0.2, 0.8, 0.2),
                xaxt = "n")
  title(xlab = "Latitude [°N]", line = 2)
  axis(1, at = seq(0, 60, by = 20))
  lmres <- lm(y~x+I(x^2), north)
  xx <- seq(min(x), max(x), by = 1)
  yy <- predict.lm(lmres, newdata = list(x = xx))
  points(xx, yy, type = "l", col = "indianred", lwd = 5)
  r2 <- signif(summary(lmres)$adj.r.squared, 3)
  text(xr, yr, bquote(r^2 == .(r2)), cex = 1.5)
}
```

```
par(mfrow = c(1,2))
myplot(north, "PCV", 20, 3.5)
myplot(north, "KCV", 30, 0.1)
```



5 Figure 3

The function `lmp()` is an utility to extract the p-value from a linear model.

```r
lmp <- function (modelobject) {
    if (class(modelobject) != "lm")
      stop("Not an object of class 'lm' ")
    f <- summary(modelobject)$fstatistic
    p <- pf(f[1], f[2], f[3], lower.tail = FALSE)
    attributes(p) <- NULL
    return(p)
}
```

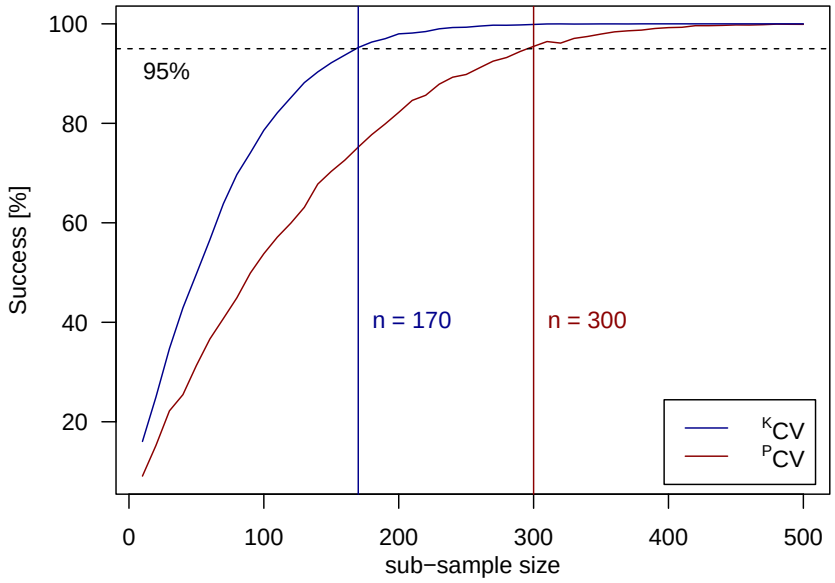
```r
manipPCV <- function(n){
  lmres <- lm(est.y~Latitude+I(Latitude^2),
              north[sample(1:nrow(north), n), ])
  return(lmp(lmres))
}
manipKCV <- function(n){
  lmres <- lm(est.x~Latitude+I(Latitude^2),
              north[sample(1:nrow(north), n), ])
  return(lmp(lmres))
}

ppvPCV <- function(n, R = 1000, alpha = 0.05){
  vectp <- replicate(R, manipPCV(n))
  return(100*sum(vectp < alpha)/R)
}
ppvKCV <- function(n, R = 1000, alpha = 0.05){
  vectp <- replicate(R, manipKCV(n))
  return(100*sum(vectp < alpha)/R)
}
```

```r
nseq <- seq(10, 500, by = 10)
lseq <- length(nseq)
resPCV <- resKCV <- numeric(lseq)
R <- 10000
for(i in 1:lseq){
  resPCV[i] <- ppvPCV(nseq[i], R = R)
  resKCV[i] <- ppvKCV(nseq[i], R = R)
}
save(nseq, resPCV, resKCV, R, file = "batch/subsampling.Rda")
```

```r
load("batch/subsampling.Rda")
par(mar = c(5, 6, 0, 1) + 0.1, pty = "m")
plot(nseq, resPCV, type = "l", col = "darkred", las = 1,
     main = "",
     xlab = "",
     ylab = "Success [%]")
title(xlab = "sub-sample size", line = 2)
points(nseq, resKCV, type = "l", col = "darkblue")
abline(h = 95, lty = 2)
```

```
legend("bottomright", inset = 0.02,
       legend = c(expression(phantom(0)^K*CV),
                  expression(phantom(0)^P*CV)),
       col = c("darkblue", "darkred"), lty = 1)
thres <- 95
vPCV <- nseq[which(resPCV > thres)[1]]
abline(v = vPCV, col = "darkred")
text(vPCV, 40, paste("n =", vPCV),pos = 4, col = "darkred")
vKCV <- nseq[which(resKCV > thres)[1]]
abline(v = vKCV, col = "darkblue")
text(vKCV, 40, paste("n =", vKCV),pos = 4, col = "darkblue")
text(0, 90, "95%", pos = 4)
```



6 Figure 4

```
# Some basic data about time-series lengths in MASTREE+
mastreeU <- mastree[!duplicated(mastree$ID), ]
quantile(mastreeU$Length)
```

```
  0%  25%  50%  75% 100%
   1    4   10   17  233
```

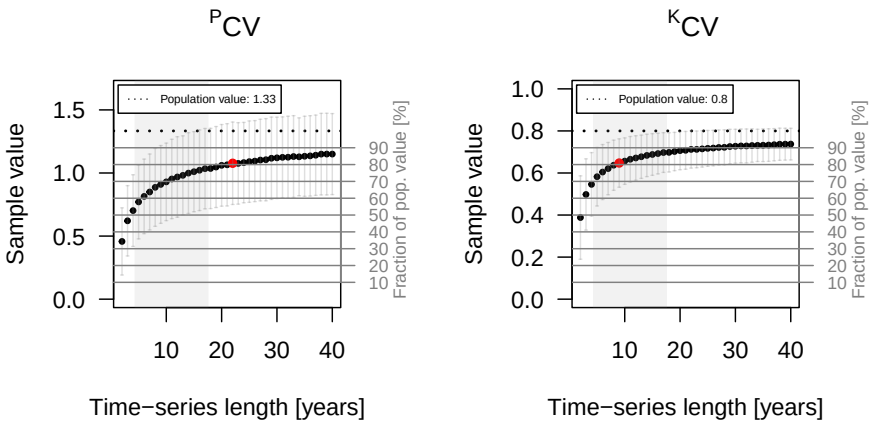
```
kcv2sig <- function(x) sqrt(-log(1-x^2))
expKCV <- function(sigma) sqrt(1 - exp(-sigma^2))
```

```
expPCV <- function(sigma) sqrt(exp(sigma^2) - 1)
tseq <- 2:40 ; nts <- length(tseq)
simPCV <- as.data.frame(matrix(NA, nrow = nts, ncol = 3))
colnames(simPCV) <- c("est", "inf", "sup")
rownames(simPCV) <- tseq
simKCV <- simPCV
R <- 10000
set.seed(1)
for(i in seq_len(nts)){
  manip <- function(){
    x <- rlnorm(tseq[i], sdlog = kcv2sig(0.8))
    return(c(pcv = PCV(x), kcv = KCV(x)))
  }
  res <- as.data.frame(t(replicate(R, manip())))
  mpcv <- mean(res$pcv) ; sdpcv <- sd(res$pcv)
  simPCV[i, "est"] <- mpcv
  simPCV[i, "inf"] <- mpcv - sdpcv
  simPCV[i, "sup"] <- mpcv + sdpcv
  mkcv <- mean(res$kcv) ; sdkcv <- sd(res$kcv)
  simKCV[i, "est"] <- mkcv
  simKCV[i, "inf"] <- mkcv - sdkcv
  simKCV[i, "sup"] <- mkcv + sdkcv
}
save(simPCV, simKCV, tseq, nts, R, expKCV, expPCV,
     file = "batch/simlnorm.Rda")
```

```
load("batch/simlnorm.Rda")
kcv2sig <- function(x) sqrt(-log(1-x^2))
myplot3 <- function(stat){
  if(stat == "PCV"){
    sim <- simPCV
    pop <- expPCV(kcv2sig(0.8))
    main <- expression(phantom(0)^P*CV)
  } else {
    sim <- simKCV
    pop <- 0.8
    main <- expression(phantom(0)^K*CV)
  }
  plot(tseq, sim[ , 1], ylim = c(0, 1.25*pop), las = 1, pch = 19,
     main = main, cex = 0.5,
     xlab = "Time-series length [years]",
     ylab = "Sample value")
  i80 <- which(sim$est/pop > 0.8)[1]
  points(tseq[i80], sim[i80, 1], pch = 19, col = "red",
         cex = 0.75)
  abline(h = pop, lty = 3, lwd = 2)
  for(i in seq_len(nts))
    myarrows(tseq[i], sim[i, "inf"], tseq[i], sim[i, "sup"])
  mygrey <- rgb(0.5, 0.5, 0.5, 0.1)
```

```
#  lines(tseq, (sim[ , "sup"] - sim[ , "inf"])/2)
  rect(4.4, -0.2, 17.5, 2, col = mygrey, border = mygrey)
  fracseq <- seq(pop/10, 9*pop/10, by = pop/10)
  mygrey2 <- grey(0.5)
  abline(h = fracseq, col = mygrey2)
  legend("topleft", inset = 0.02, cex = 0.5,
       legend = paste("Population value:", signif(pop, 3)),
       lty = 3, bg = "white")
  axis(4, at = fracseq, labels = 10*(1:9), las = 1, col = mygrey2,
       col.axis = mygrey2, cex.axis = 0.75)
  text(50, 0, "Fraction of pop. value [%]", srt = 90,
        col = mygrey2, xpd = NA, pos = 4, cex = 0.75)
}

par(mfrow = c(1, 2), mar = c(5, 4, 4, 4) + 0.1, pty = "s")
myplot3("PCV")
myplot3("KCV")
```



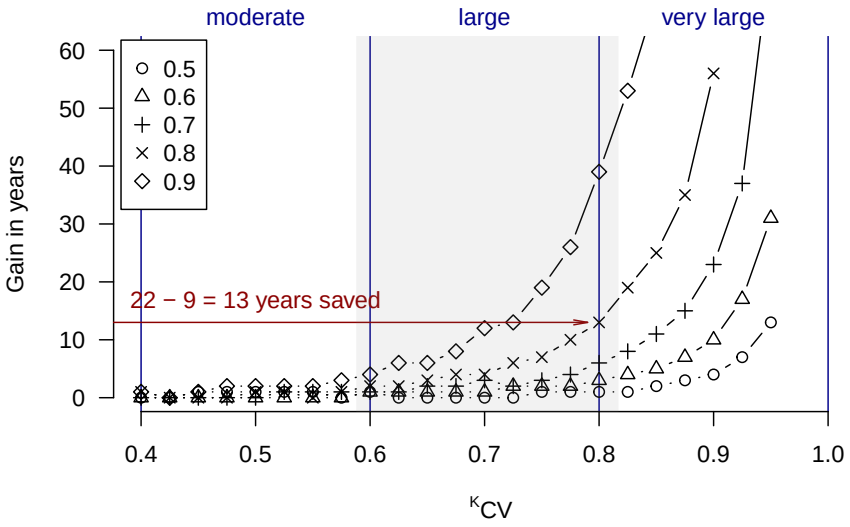
7 Figure 5

```
kvaseq <- seq(0.4, 0.95, by = 0.025) ; nkva <- length(kvaseq)
sdlogseq <- kcv2sig(kvaseq)
saved <- vector("list", nkva)
R <- 10000
set.seed(1)
```

```
for(j in seq_len(nkva)){
  tseq <- 2:100 ; nts <- length(tseq)
  simPCV <- as.data.frame(matrix(NA, nrow = nts, ncol = 3))
  colnames(simPCV) <- c("est", "inf", "sup")
  rownames(simPCV) <- tseq
  simKCV <- simPCV
  for(i in seq_len(nts)){
    manip <- function(){
      x <- rlnorm(tseq[i], sdlog = sdlogseq[j])
      return(c(pcv = PCV(x), kcv = KCV(x)))
    }
    res <- as.data.frame(t(replicate(R, manip())))
    mpcv <- mean(res$pcv) ; sdpcv <- sd(res$pcv)
    simPCV[i, "est"] <- mpcv
    simPCV[i, "inf"] <- mpcv - sdpcv
    simPCV[i, "sup"] <- mpcv + sdpcv
    mkcv <- mean(res$kcv) ; sdkcv <- sd(res$kcv)
    simKCV[i, "est"] <- mkcv
    simKCV[i, "inf"] <- mkcv - sdkcv
    simKCV[i, "sup"] <- mkcv + sdkcv
  }
  saved[[j]] <- list(simPCV = simPCV, simKCV = simKCV)
}
save(kvaseq, sdlogseq, R, saved, file = "batch/savedyears.Rda")
```

```
load("batch/savedyears.Rda")
gain <- function(simPCV, simKCV, frac, sigma = 1){
  yPCV <- rownames(simPCV)[which(simPCV$est/expPCV(sigma) > frac)[1]]
  yKCV <- rownames(simKCV)[which(simKCV$est/expKCV(sigma) > frac)[1]]
  return(as.numeric(yPCV) - as.numeric(yKCV))
}
mygrey <- rgb(0.5, 0.5, 0.5, 0.1)
fracseq <- seq(0.5, 0.9, by = 0.1)
nfrac <- length(fracseq)
nkva <- length(kvaseq)
y <- numeric(nkva)
plot.new() ; plot.window(xlim = c(0.4, 1), ylim = c(0, 60))
text(KvalsethBreaks[-c(1:3)] - 0.1, 60+2,
     KvalsethVerb[-c(1:2)], col = "darkblue", xpd = NA, pos = 3)
abline(v = KvalsethBreaks, col = "darkblue")
axis(1, at = seq(0.4, 1, by = 0.1)) ; axis(2, las = 1)
title(ylab = "Gain in years", xlab = expression(phantom(0)^K*CV))
for(f in seq_len(nfrac)){
  for(i in seq_len(nkva)){
    simPCV <- saved[[i]]$simPCV
    simKCV <- saved[[i]]$simKCV
    y[i] <- gain(simPCV, simKCV, fracseq[f], sdlogseq[i])
  }
  points(kvaseq, y, type = "b", pch = f)
```

```
#  text(kvaseq, y, y, pos = 4)
}
legend("topleft", inset = 0.01, legend = fracseq, pch = 1:nfrac,
       bg = "white")
arrows(0.3, 13, 0.79, 13, col = "darkred", angle =  10, le = 0.1)
text(0.5, 13, "22 - 9 = 13 years saved", pos = 3, col = "darkred")
qts <- quantile(eisKCV$est)
rect(qts[2], -10, qts[4], 70, col = mygrey, border = mygrey)
```



8 References

Allaire, JJ, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. 2020. *Rmarkdown: Dynamic Documents for r*. https://github.com/rstudio/rmarkdown.

Canty, A., and B. D. Ripley. 2021. *Boot: Bootstrap R (S-Plus) Functions*.

Davison, D. V., A. C. Hinkley. 1997. *Bootstrap Methods and Their Applications*. Cambridge: Cambridge University Press.

Efron, B. 1987. "Better Bootstrap Confidence Intervals." *Journal of the American Statistical Association* 82 (397): 171–85.

Hacket-Pain, A., J. J. Foest, I. S. Pearse, J. M. LaMontagne, W. D. Koenig, G. Vacchiano, M. Bogdziewicz, et al. 2022. "MASTREE+: Time-Series of Plant Reproductive Effort from Six Continents." *Global Change Biology* 00: 1–17.

Heath, J. P. 2006. "Quantifying Temporal Variability in Population Abundances." *Oikos* 115 (3): 573–81.

Heath, J. P., and P. Borowski. 2013. "Quantifying Proportional Variability." *PLoS One* 8 (12): e84074.

Kvålseth, T. O. 2017. "Coefficient of Variation: The Second-Order Alternative." *Journal of Applied Statistics* 44: 402–15.

Pearson, K. 1896. "VII. Mathematical Contributions to the Theory of Evolution.—III. Regression, Heredity, and Panmixia." *Philosophical Transactions of the Royal Society of London. Series A*, no. 187: 253–318.

Xie, Yihui. 2014. "Knitr: A Comprehensive Tool for Reproducible Research in R." In *Implementing Reproducible Computational Research*, edited by Victoria Stodden, Friedrich Leisch, and Roger D. Peng. Chapman; Hall/CRC. http://www.crcpress.com/product/isbn/9781466561595.

———. 2015. *Dynamic Documents with R and Knitr*. 2nd ed. Boca Raton, Florida: Chapman; Hall/CRC. https://yihui.org/knitr/.

———. 2019. "TinyTeX: A Lightweight, Cross-Platform, and Easy-to-Maintain LaTeX Distribution Based on TeX Live." *TUGboat*, no. 1: 30–32. http://tug.org/TUGboat/Contents/contents40-1.html.

———. 2020a. *Knitr: A General-Purpose Package for Dynamic Report Generation in r*. https://yihui.org/knitr/.

———. 2020b. *Tinytex: Helper Functions to Install and Maintain TeX Live, and Compile LaTeX Documents*. https://github.com/yihui/tinytex.

Xie, Yihui, J. J. Allaire, and Garrett Grolemund. 2018. *R Markdown: The Definitive Guide*. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown.

Xie, Yihui, Christophe Dervieux, and Emily Riederer. 2020. *R Markdown Cookbook*. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown-cookbook.

9 Annexes

9.1 Session informations

- R version 4.2.1 (2022-06-23), `x86_64-apple-darwin17.0`
- Locale: `C/C/C/C/C/fr_FR.UTF-8`
- Running under: `macOS Big Sur ... 10.16`
- Matrix products: default
- BLAS:
  `/Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRblas.0.dylib`
- LAPACK:
  `/Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRlapack.dylib`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: boot 1.3-28
- Loaded via a namespace (and not attached): cli 3.4.1, compiler 4.2.1, digest 0.6.29, evaluate 0.16, fastmap 1.1.0, glue 1.6.2, highr 0.9, htmltools 0.5.3, knitr 1.40, lifecycle 1.0.3, magrittr 2.0.3, parallel 4.2.1, rlang 1.0.6, rmarkdown 2.16, rstudioapi 0.14, stringi 1.7.8, stringr 1.5.0, tools 4.2.1, vctrs 0.5.2, xfun 0.33, yaml 2.3.5

9.2 LaTeX code used

```
%%%%%%%%%%%%%%%%%%%%%%%%
% LaTeX packages used %
%%%%%%%%%%%%%%%%%%%%%%%%

\usepackage[english]{babel}
\usepackage{lettrine}
\usepackage{verbatim}

%%%%%%%%%%%%%%%%%%
% LaTeX macros %
%%%%%%%%%%%%%%%%%%

\newcommand{\myinput}[1]{
  \scriptsize
  \medskip
  \VerbatimInput[frame=single,label=#1]{#1}
  \normalsize
}

\newcommand{\Rlogo}{
```

```
  \protect
  \includegraphics[height=1.6ex,keepaspectratio]{figs/Rlogo.pdf}
}

\newcommand{\refs}[1]{
  section~\ref{#1} page~\pageref{#1}
}

\newcommand{\reff}[1]{
  figure~\ref{#1} page~\pageref{#1}
}

\newcommand{\reft}[1]{
  table~\ref{#1} page~\pageref{#1}
}

\newcommand{\myjpg}[3]{
  \begin{figure}
    \begin{center}
      \fbox{\begin{minipage}{#2\textwidth}
        \includegraphics[width=\textwidth]{figs/#1.jpg}
        \caption{\label{#1} #3}
      \end{minipage}}
    \end{center}
  \end{figure}
}
```