Simulationen mit SAS und R
ein Vergleich

Karl Moder

Institute für Angewandte Statistik und EDV
Department für Raum, Landschaft und Infrastruktur
Universität für Bodenkultur Wien

2.10.2009

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Wichtige Strukturen bei Simulation und Analyse

- FOR (WHILE, REPEAT) / DO Schleifen
- IF Blöcke
- Wertzuweisungen
- Addition

Wichtige Strukturen bei Simulation und Analyse

- FOR (WHILE, REPEAT) / DO Schleifen
- IF Blöcke
- Wertzuweisungen
- Addition

Wichtige Strukturen bei Simulation und Analyse

- FOR (WHILE, REPEAT) / DO Schleifen
- IF Blöcke
- Wertzuweisungen
- Addition

Wichtige Strukturen bei Simulation und Analyse

- FOR (WHILE, REPEAT) / DO Schleifen
- IF Blöcke
- Wertzuweisungen
- Addition

Funktionen zur Zeitmessung in R und SAS

R	SAS

Funktionen zur Zeitmessung in R und SAS

R	SAS
system.time(expr, gcFirst) • expr: Ausdruck für den die Zeit gemessen werden soll • gcFirst: TRUE/FALSE soll eine Speicherbereinigung durchgeführt werden	

Funktionen zur Zeitmessung in R und SAS

R

system.time(expr, gcFirst)

- expr: Ausdruck für den die Zeit gemessen werden soll
- gcFirst: TRUE/FALSE soll eine Speicherbereinigung durchgeführt werden

SAS

Ergebnis

user:
CPU-Zeit für die Ausführung des Benutzerbefehls

Funktionen zur Zeitmessung in R und SAS

R

system.time(expr, gcFirst)

- expr: Ausdruck für den die Zeit gemessen werden soll
- gcFirst: TRUE/FALSE soll eine Speicherbereinigung durchgeführt werden

SAS

Ergebnis

user:
CPU-Zeit für die Ausführung des Benutzerbefehls

system:
CPU-Zeit die für Aufrufe von Betriebssystembefehlen

Funktionen zur Zeitmessung in R und SAS

R	SAS
system.time(expr, gcFirst) • expr: Ausdruck für den die Zeit gemessen werden soll • gcFirst: TRUE/FALSE soll eine Speicherbereinigung durchgeführt werden	

Ergebnis

user: CPU-Zeit für die Ausführung des Benutzerbefehls system: CPU-Zeit die für Aufrufe von Betriebssystembefehlen total elapsed times: Gesamtzeit für die Ausführung der Instruktionen

Funktionen zur Zeitmessung in R und SAS

R	SAS
system.time(expr, gcFirst) • expr: Ausdruck für den die Zeit gemessen werden soll • gcFirst: TRUE/FALSE soll eine Speicherbereinigung durchgeführt werden	options fullstimer

Ergebnis

user:
CPU-Zeit für die Ausführung des Benutzerbefehls

system:
CPU-Zeit die für Aufrufe von Betriebssystembefehlen

total elapsed times:
Gesamtzeit für die Ausführung der Instruktionen

Funktionen zur Zeitmessung in R und SAS

R	SAS
system.time(expr, gcFirst) • expr: Ausdruck für den die Zeit gemessen werden soll • gcFirst: TRUE/FALSE soll eine Speicherbereinigung durchgeführt werden	options fullstimer

Ergebnis

user: CPU-Zeit für die Ausführung des Benutzerbefehls system: CPU-Zeit die für Aufrufe von Betriebssystembefehlen total elapsed times: Gesamtzeit für die Ausführung der Instruktionen	User CPU Time: CPU-Zeit für die Ausführung des Benutzerbefehls System CPU Time: CPU-Zeit die für Aufrufe von Betriebssystembefehlen Real Time: Gesamtzeit für die Ausführung der Instruktionen

Geschwindigkeitsvergleich Schleifen

Geschwindigkeitsvergleich Schleifen

R

```
for (wh in 1:10)
+ {
+ t1=Sys.time();
+ print(system.time
+   (for (i in 1 :100000000){}));
+ print(difftime(t1,Sys.time()));
+ };
```

Geschwindigkeitsvergleich Schleifen

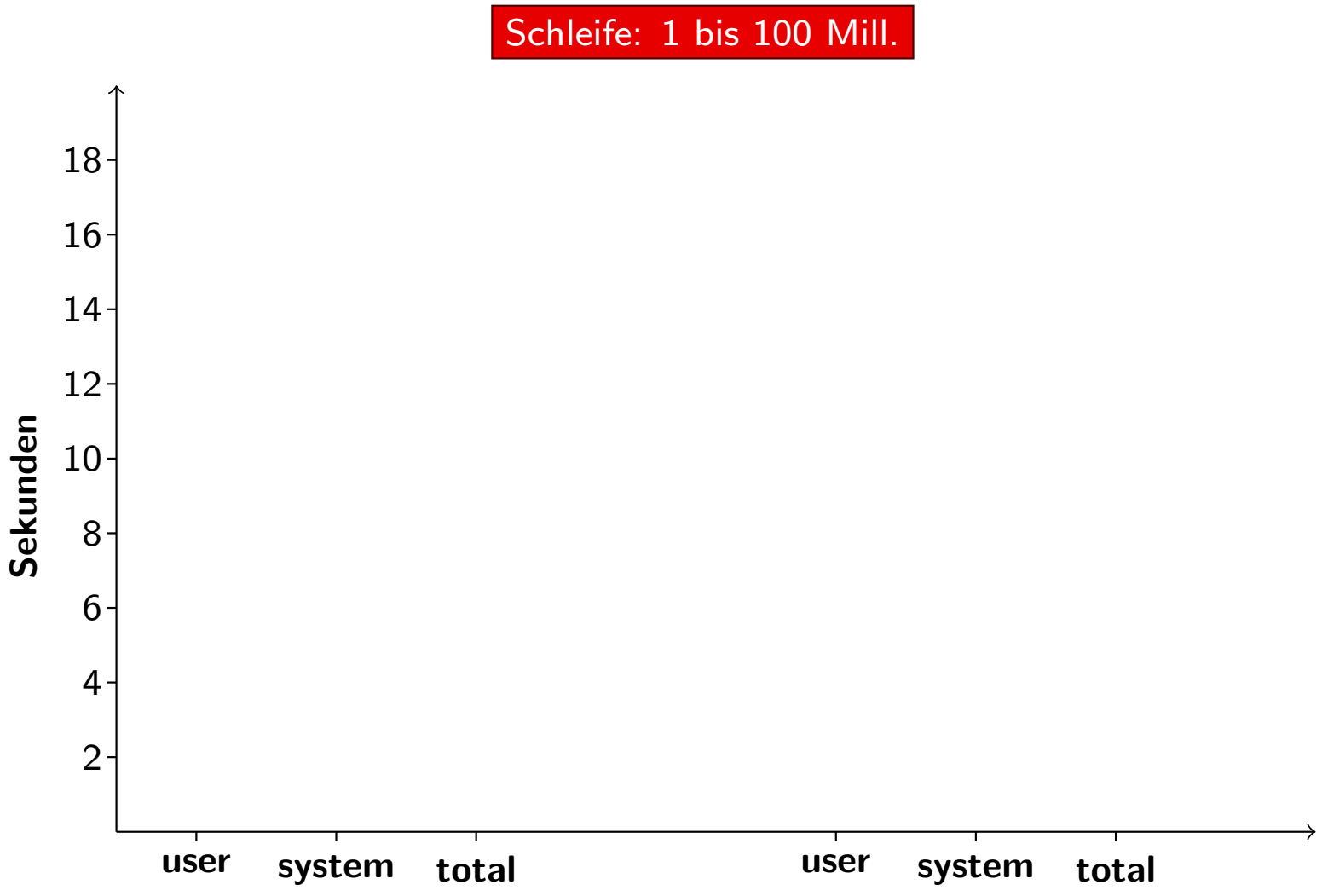
R

```
for (wh in 1:10)
+ {
+ t1=Sys.time();
+ print(system.time
+   (for (i in 1 :100000000){}));
+ print(difftime(t1,Sys.time()));
+ };
```

SAS

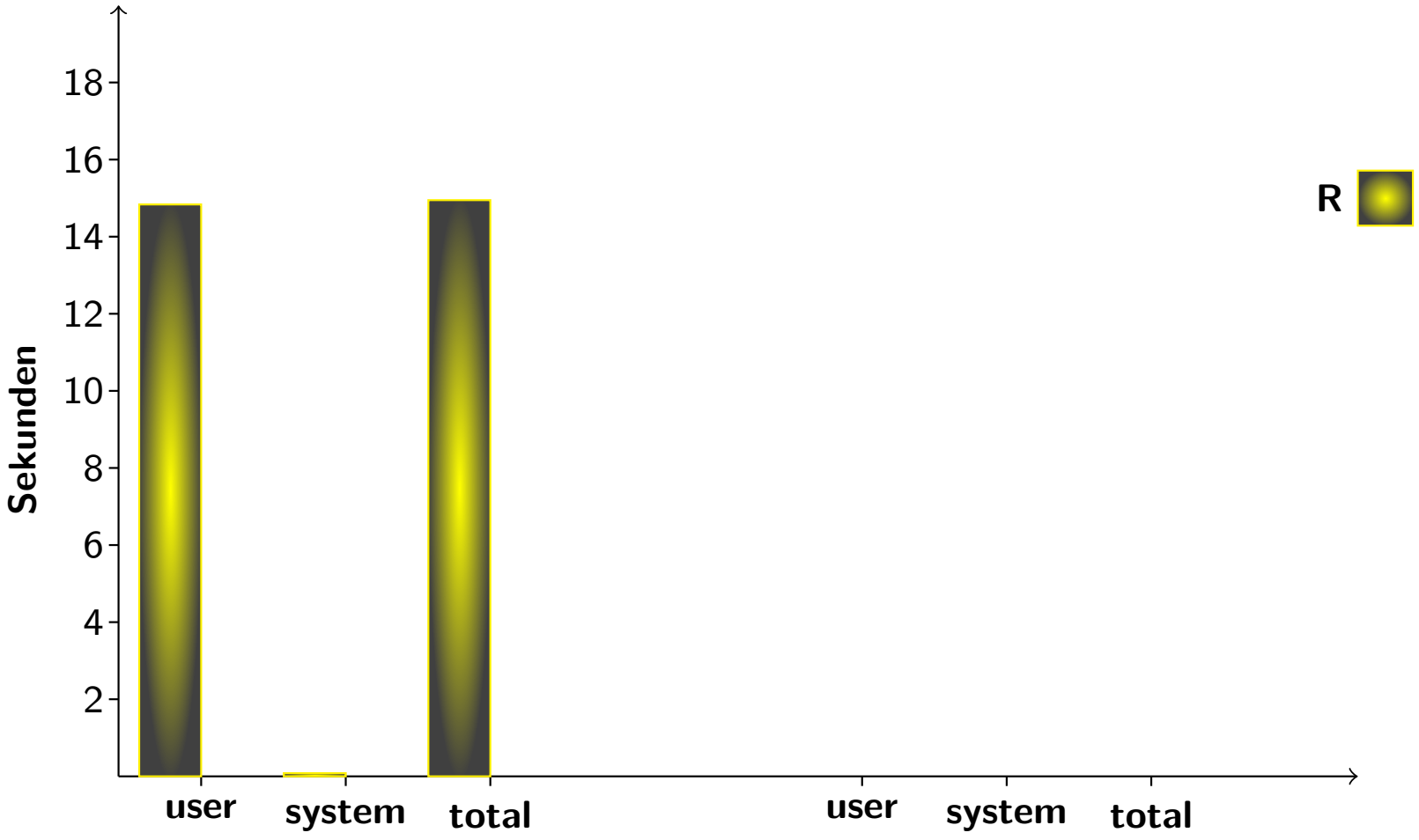
```
%macro iml1;
%do j=1 %to 10;
%let time1=%sysfunc(time());
   proc iml;
   do i=1 to 100000000; end;
   run;
%let time2=%sysfunc(time());
%let diff=%Sysevalf(&time2-&time1);
%put &diff;
%end;
%mend iml1;
run;
%iml1 run;
```

Geschwindigkeitsvergleich Schleifen



Geschwindigkeitsvergleich Schleifen

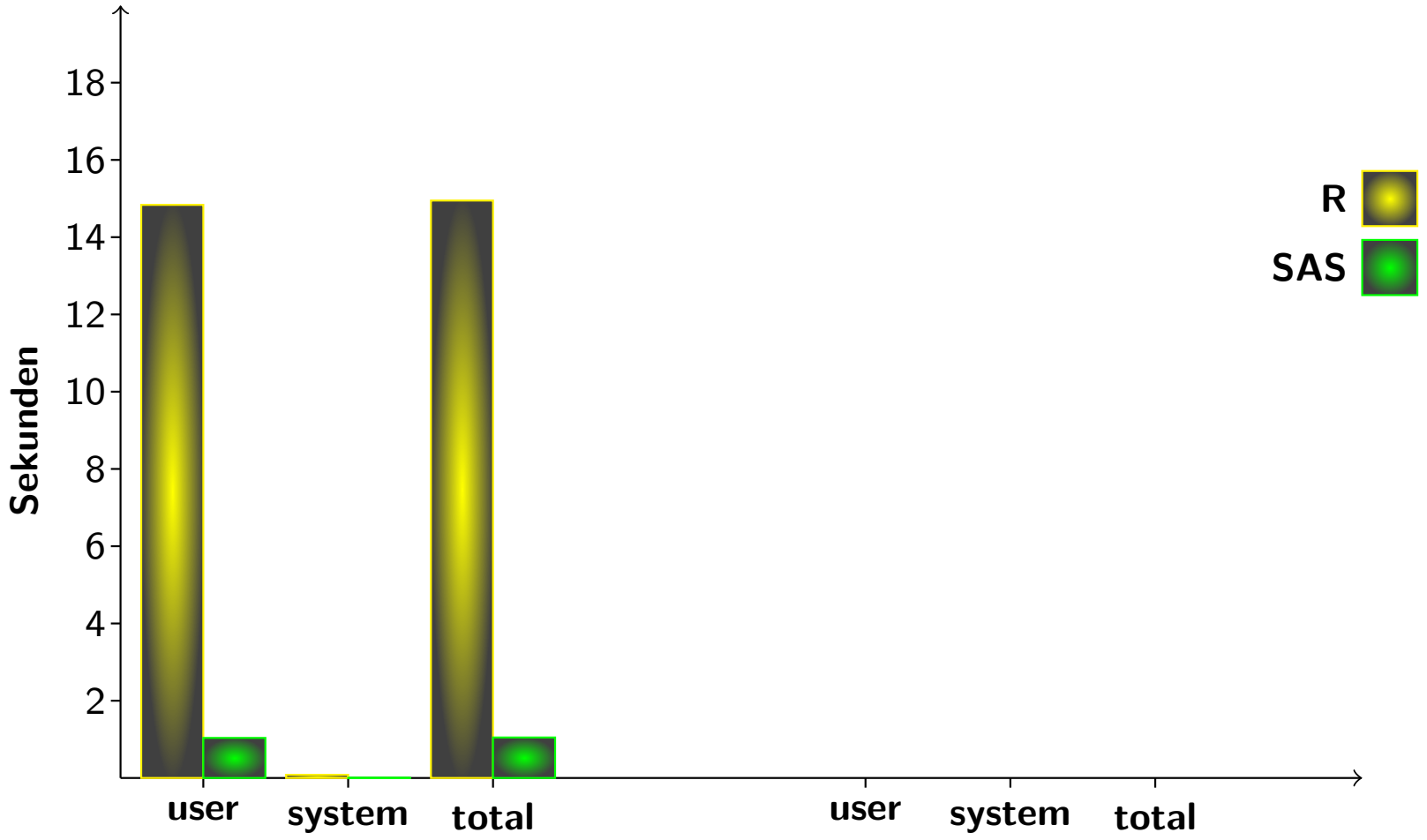
Schleife: 1 bis 100 Mill.



Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

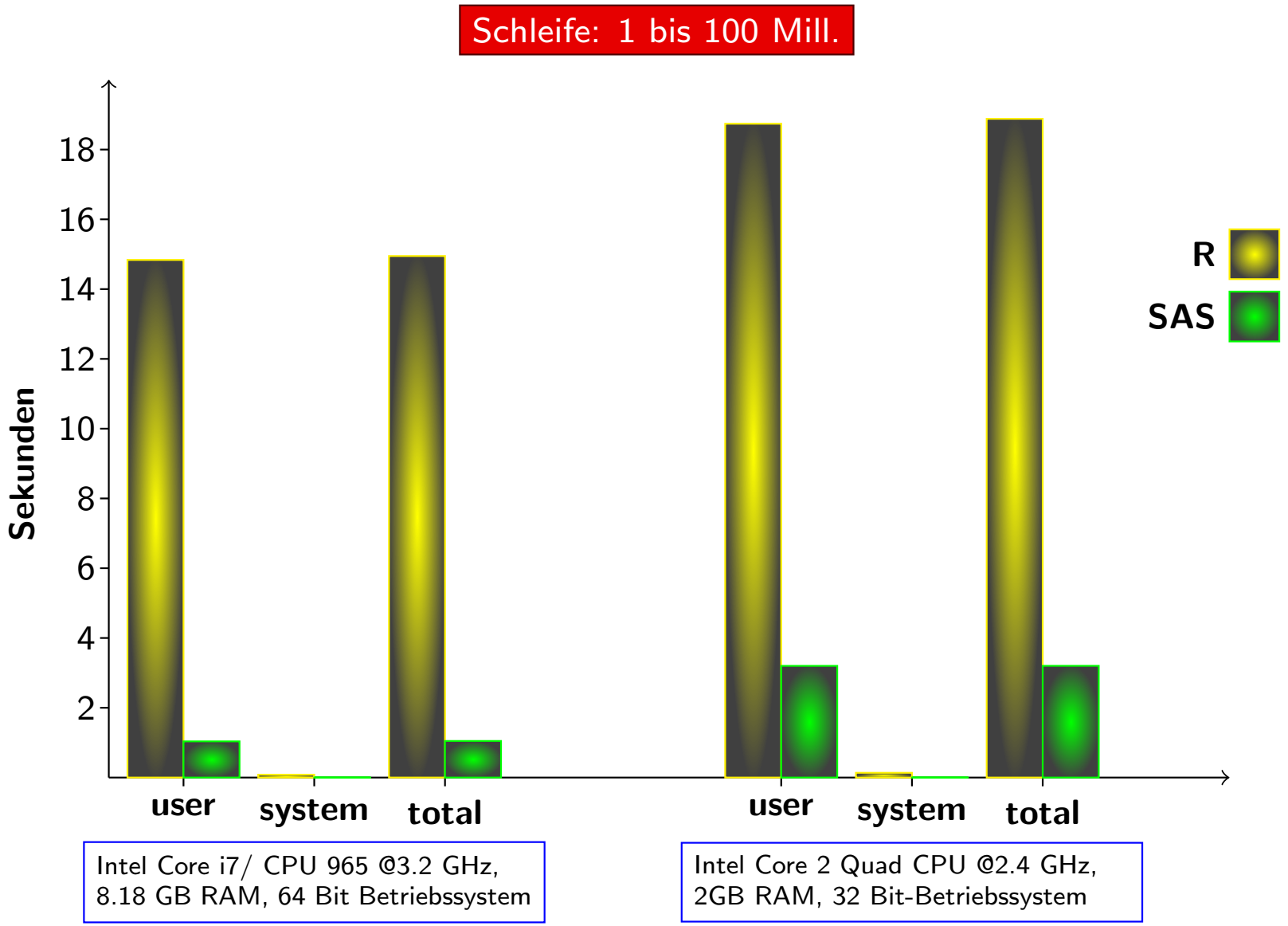
Geschwindigkeitsvergleich Schleifen

Schleife: 1 bis 100 Mill.



Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

Geschwindigkeitsvergleich Schleifen



Geschwindigkeitsvergleich IF

Geschwindigkeitsvergleich IF

R

```
for (wh in 1:10)
+ {
+ t1=Sys.time();
+ print(system.time
+   (for (i in 1 :100000000){
+      if (i==1){}
+ ));
+ print(difftime(t1,Sys.time()));
+ };
```

Geschwindigkeitsvergleich IF

R

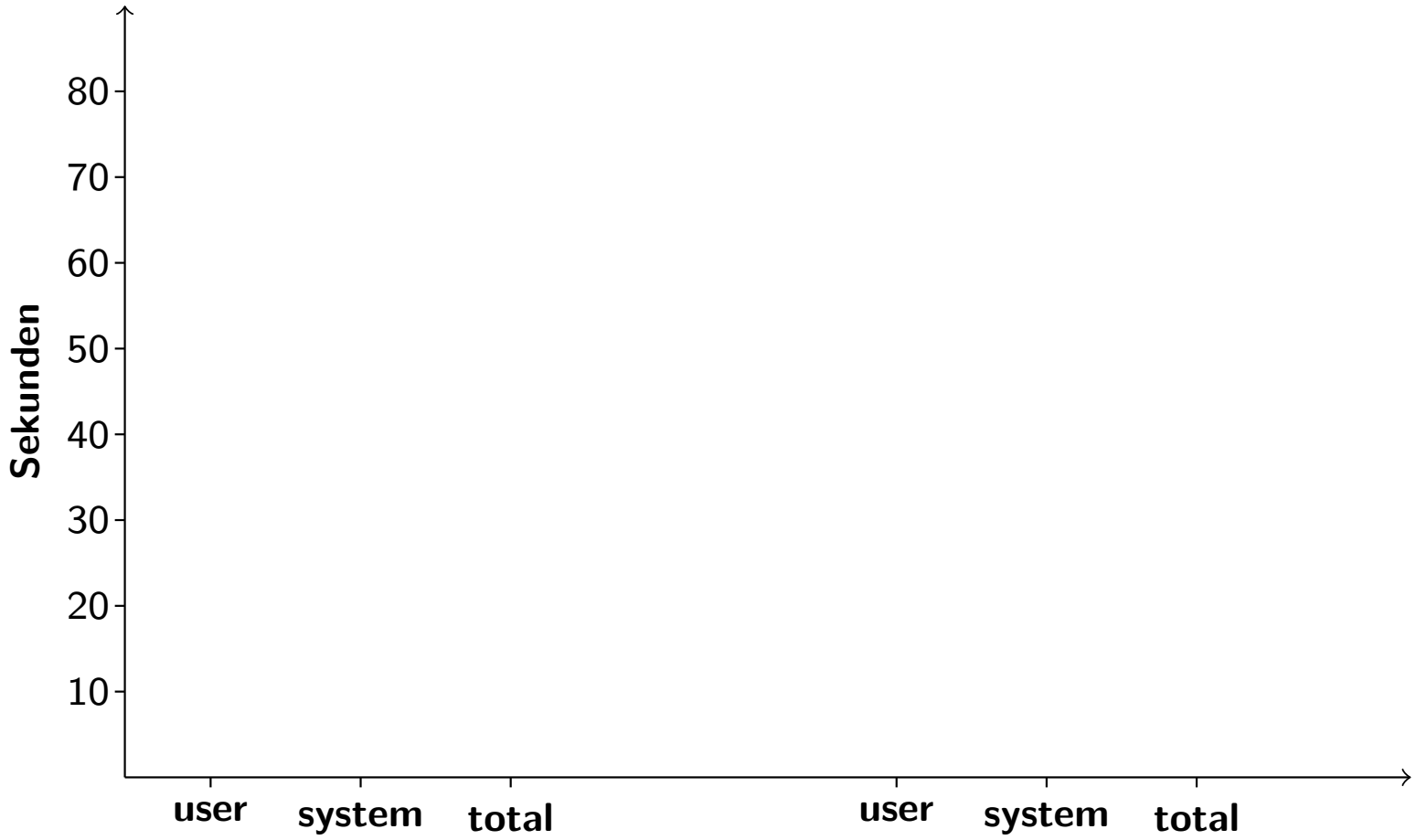
```
for (wh in 1:10)
+ {
+ t1=Sys.time();
+ print(system.time
+   (for (i in 1 :100000000){
+     if (i==1){}
+ ));
+ print(difftime(t1,Sys.time()));
+ };
```

SAS

```
%macro iml3;
%do j=1 %to 10;
%let time1=%sysfunc(time());
  proc iml;
  do i=1 to 100000000;
    if i=1 then do; end;
  end;
  run;
%let time2=%sysfunc(time());
%let diff=%Sysevalf(&time2-&time1);
%put &diff;
%end;
%mend iml3;
run;
%iml3 run;
```

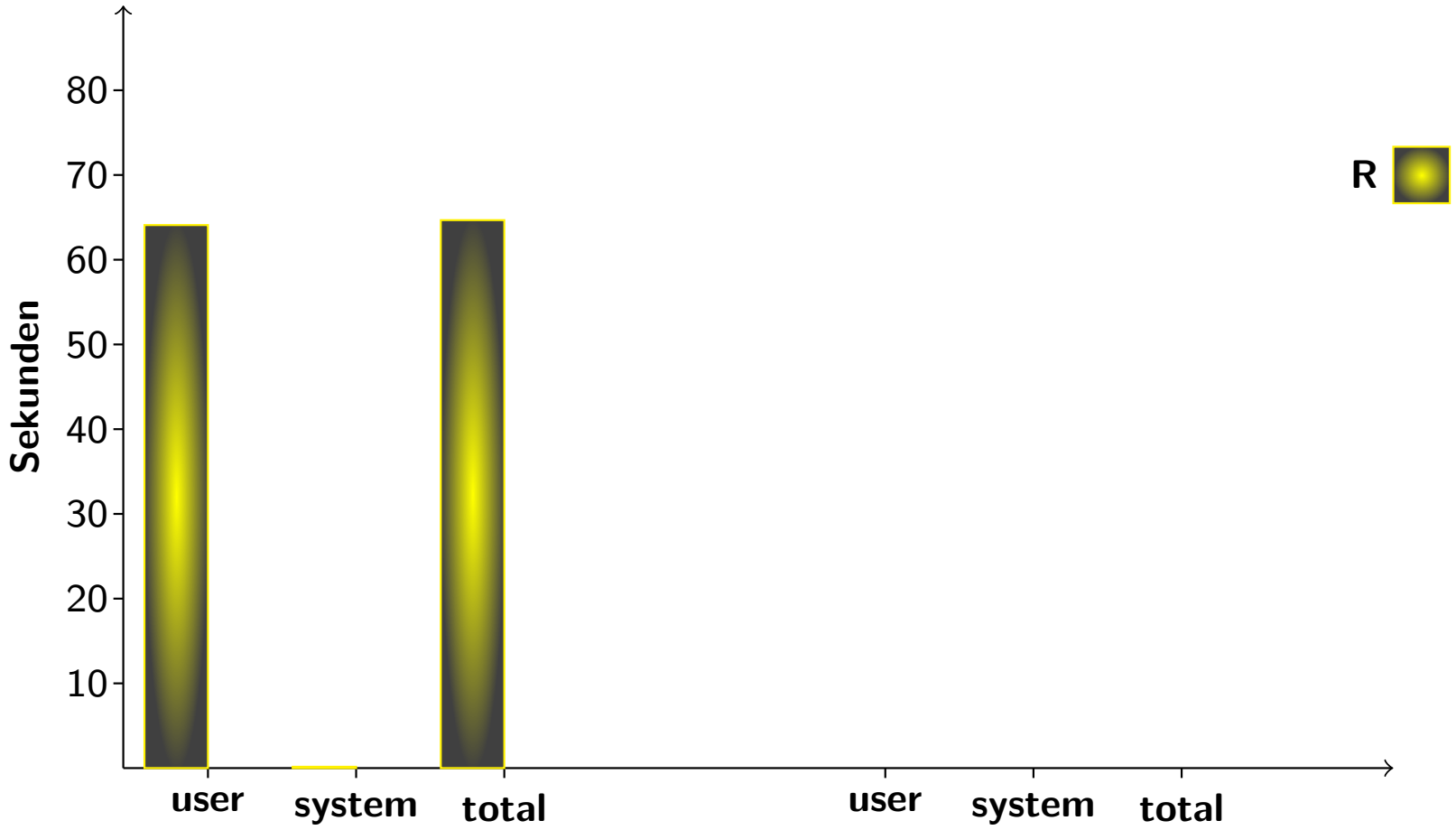
Geschwindigkeitsvergleich IF

100 Mill. Wiederholungen



Geschwindigkeitsvergleich IF

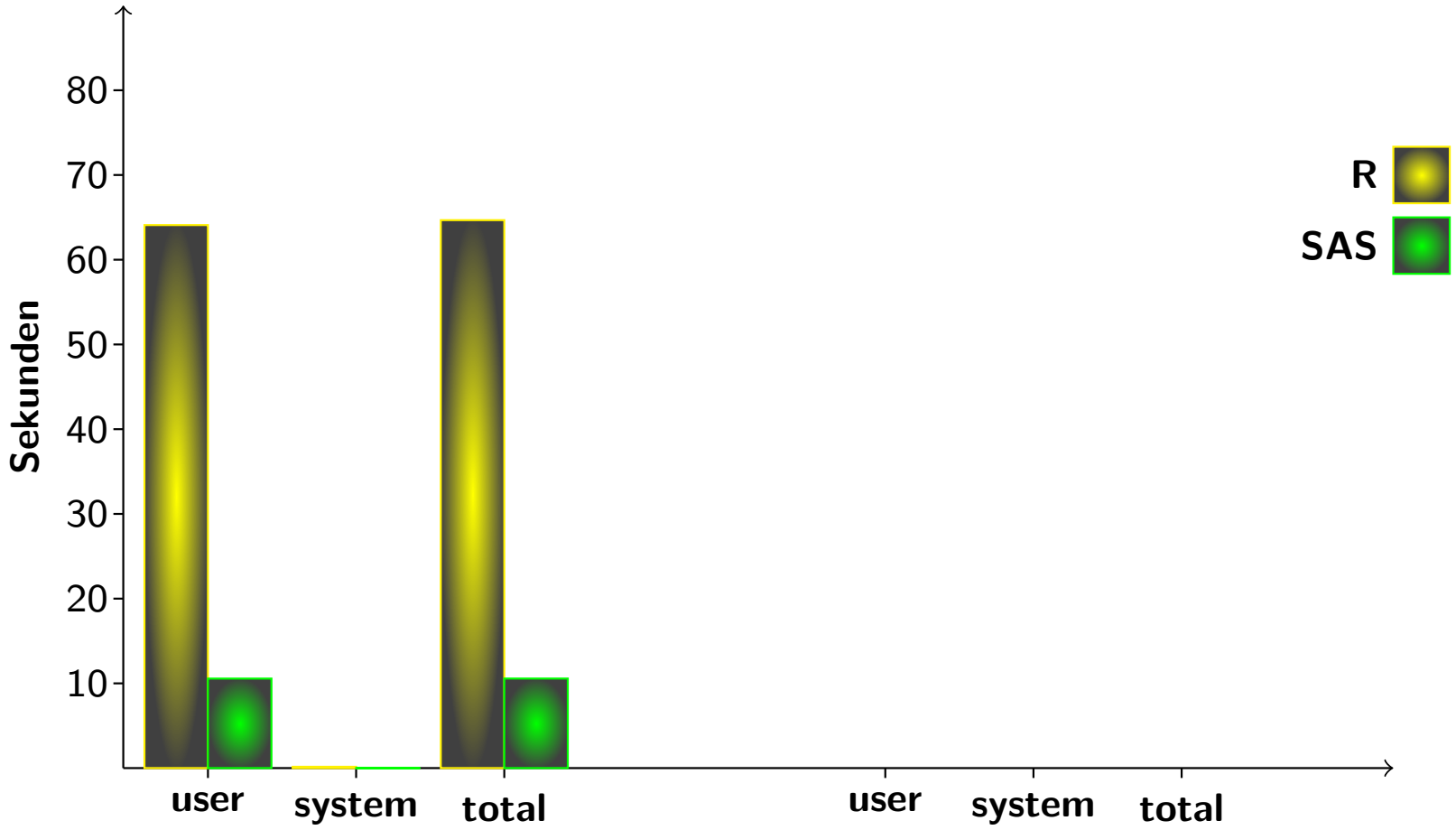
100 Mill. Wiederholungen



Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

Geschwindigkeitsvergleich IF

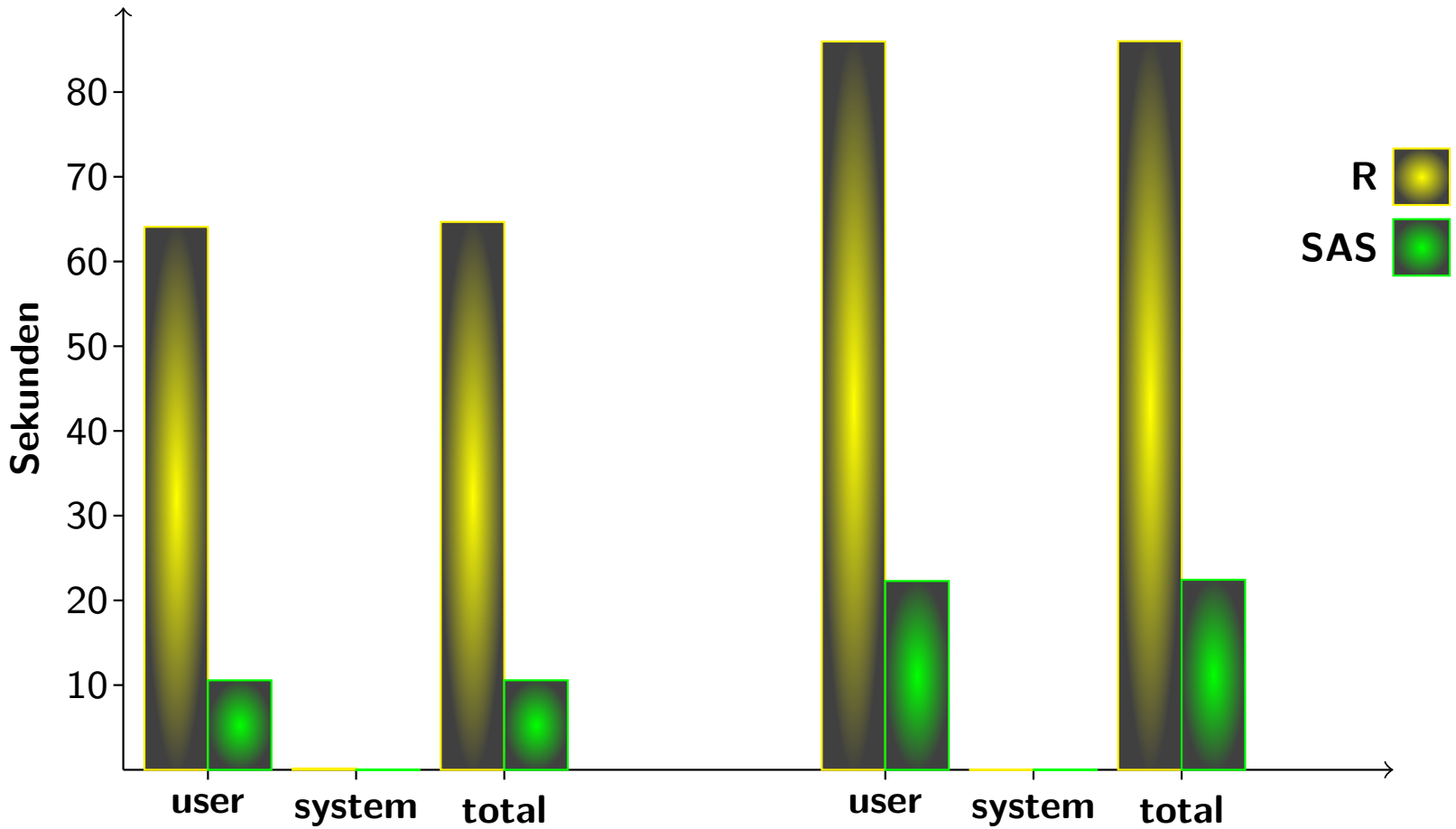
100 Mill. Wiederholungen



Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

Geschwindigkeitsvergleich IF

100 Mill. Wiederholungen



Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

Intel Core 2 Quad CPU @2.4 GHz,
2GB RAM, 32 Bit-Betriebssystem

Geschwindigkeitsvergleich Wertzuweisung

Geschwindigkeitsvergleich Wertzuweisung

R

```
for (wh in 1:10)
+ {
+ t1=Sys.time();
+ print(system.time
+   (for (i in 1 :100000000){
+      k=1;
+ }));
+ print(difftime(t1,Sys.time()));
+ };
```

Geschwindigkeitsvergleich Wertzuweisung

R

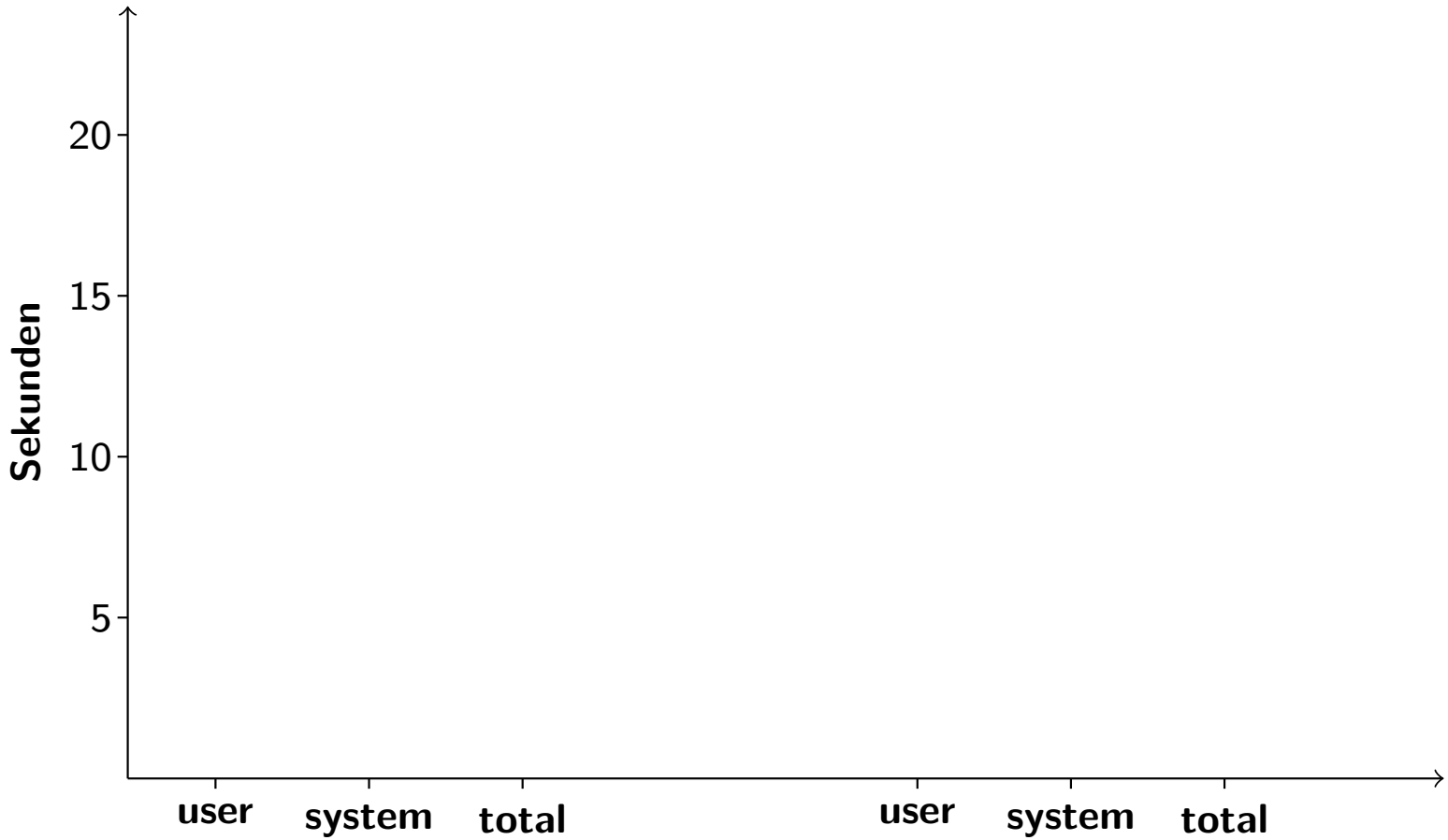
```
for (wh in 1:10)
+ {
+ t1=Sys.time();
+ print(system.time
+   (for (i in 1 :100000000){
+      k=1;
+ }));
+ print(difftime(t1,Sys.time()));
+ };
```

SAS

```
%macro iml3;
%do j=1 %to 10;
%let time1=%sysfunc(time());
   proc iml;
   do i=1 to 100000000;
      k=1;
   end;
   run;
%let time2=%sysfunc(time());
%let diff=%Sysevalf(&time2-&time1);
%put &diff;
%end;
%mend iml3;
run;
%iml3 run;
```

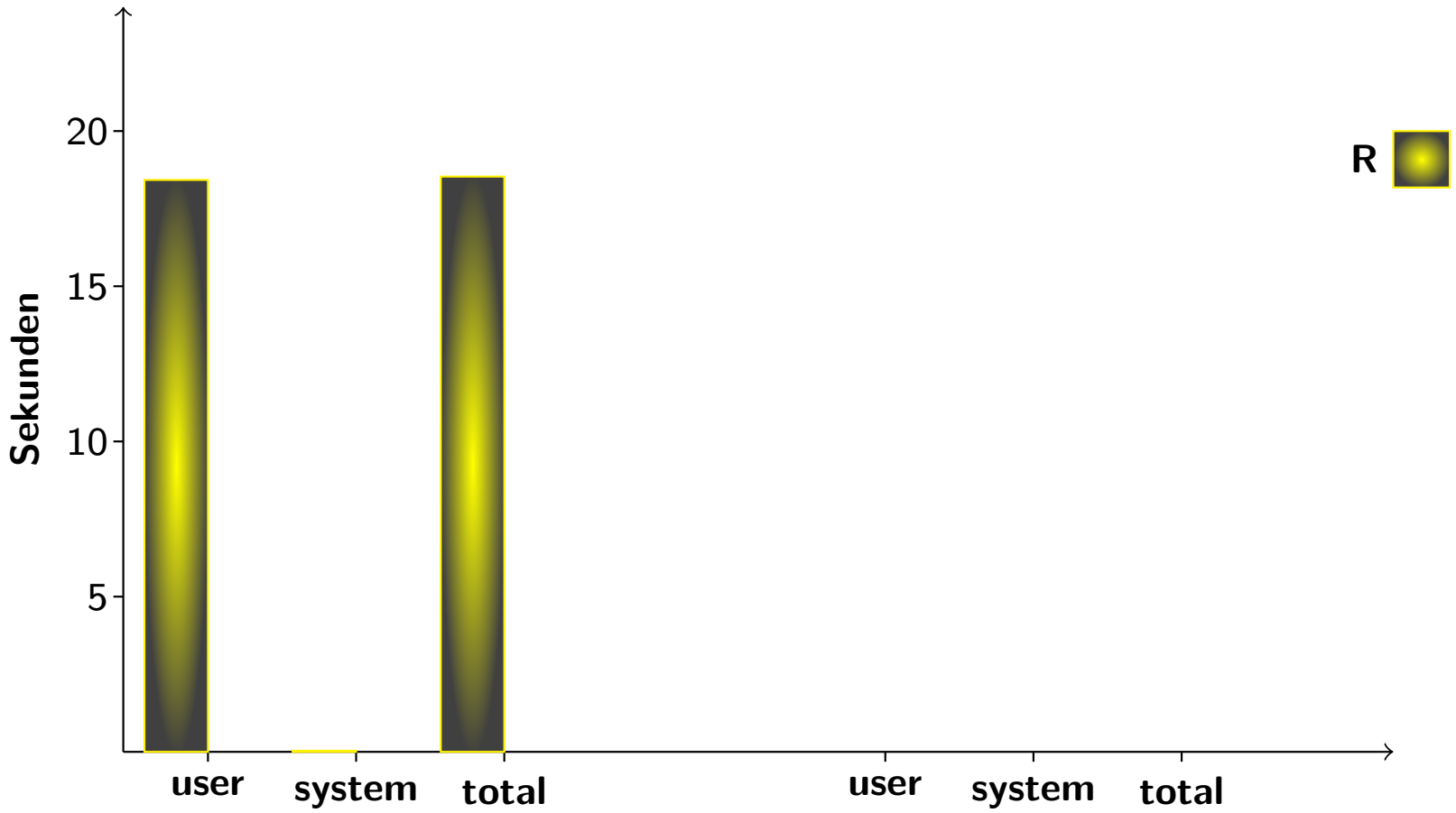
Geschwindigkeitsvergleich Wertzuweisung

100 Mill. Wiederholungen



Geschwindigkeitsvergleich Wertzuweisung

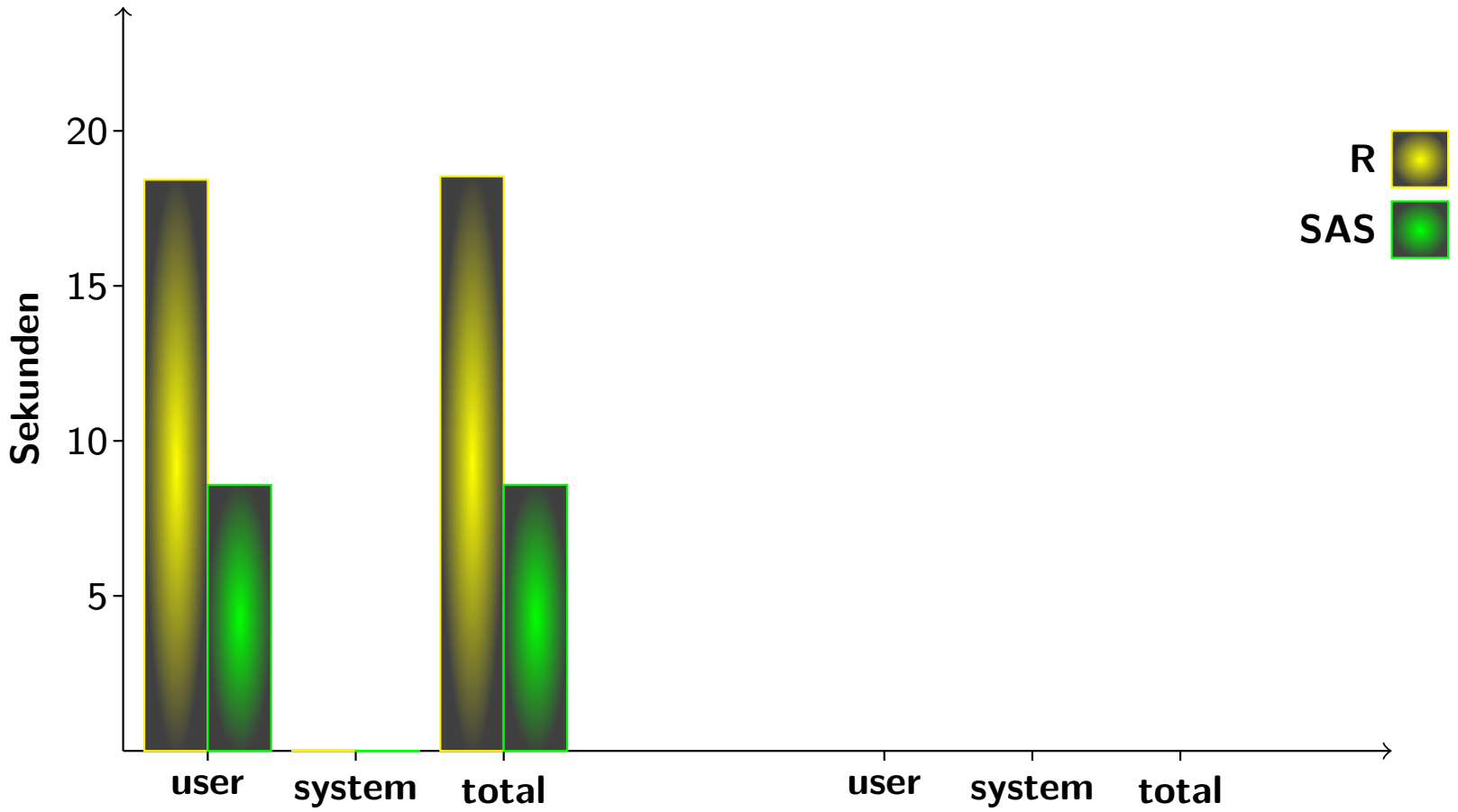
100 Mill. Wiederholungen



Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

Geschwindigkeitsvergleich Wertzuweisung

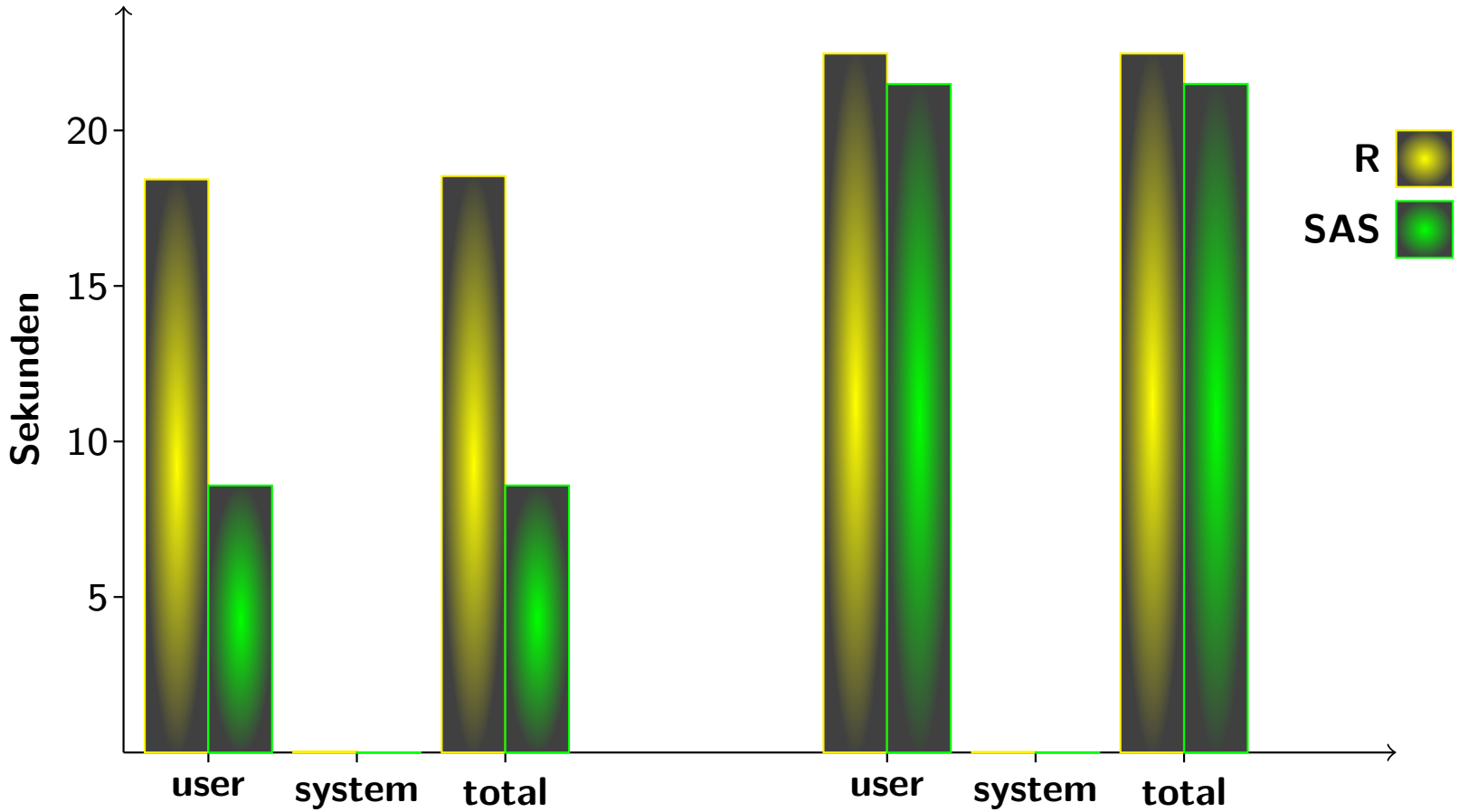
100 Mill. Wiederholungen



Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

Geschwindigkeitsvergleich Wertzuweisung

100 Mill. Wiederholungen



Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

Intel Core 2 Quad CPU @2.4 GHz,
2GB RAM, 32 Bit-Betriebssystem

Geschwindigkeitsvergleich Addition

Geschwindigkeitsvergleich Addition

R

```
for (wh in 1:10)
+ {
+ t1=Sys.time();
+ print(system.time
+   (for (i in 1 :100000000){
+      k=i+i;
+ }));
+ print(difftime(t1,Sys.time()));
+ };
```

Geschwindigkeitsvergleich Addition

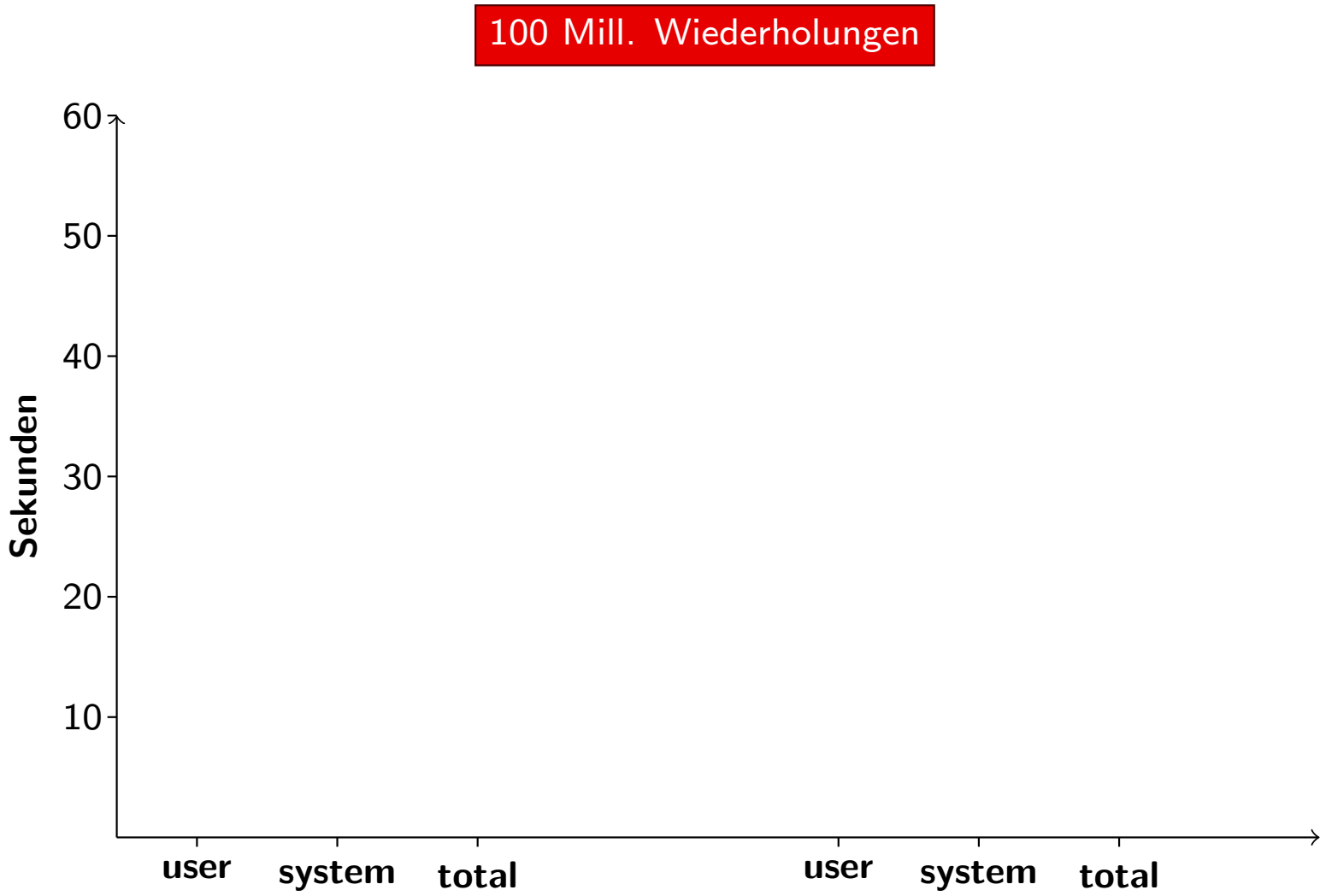
R

```
for (wh in 1:10)
+ {
+ t1=Sys.time();
+ print(system.time
+   (for (i in 1 :100000000){
+      k=i+i;
+ }));
+ print(difftime(t1,Sys.time()));
+ };
```

SAS

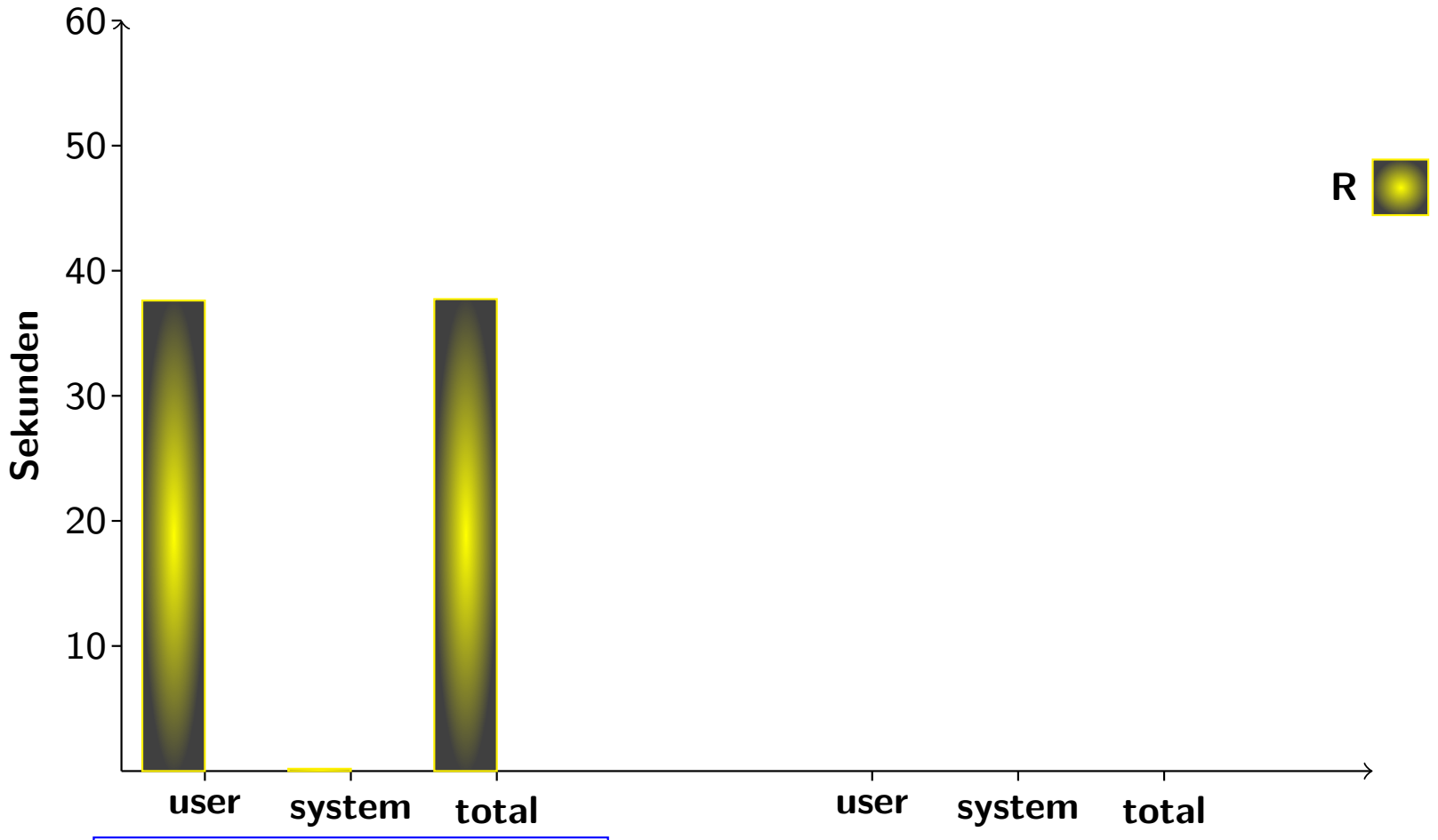
```
%macro iml3;
%do j=1 %to 10;
%let time1=%sysfunc(time());
   proc iml;
   do i=1 to 100000000;
      k=i+i;
   end;
   run;
%let time2=%sysfunc(time());
%let diff=%Sysevalf(&time2-&time1);
%put &diff;
%end;
%mend iml3;
run;
%iml3 run;
```

Geschwindigkeitsvergleich Addition



Geschwindigkeitsvergleich Addition

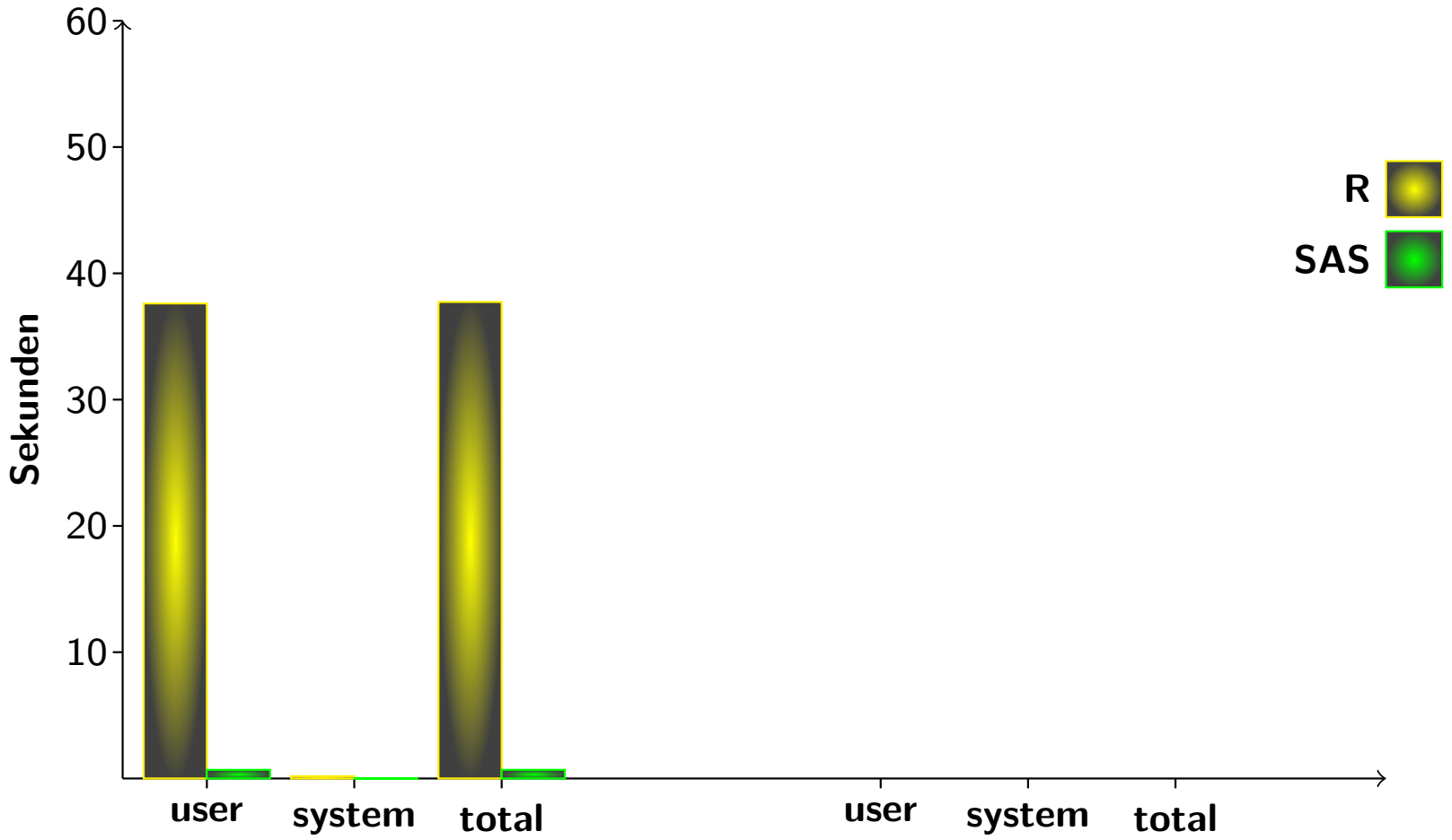
100 Mill. Wiederholungen



Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

Geschwindigkeitsvergleich Addition

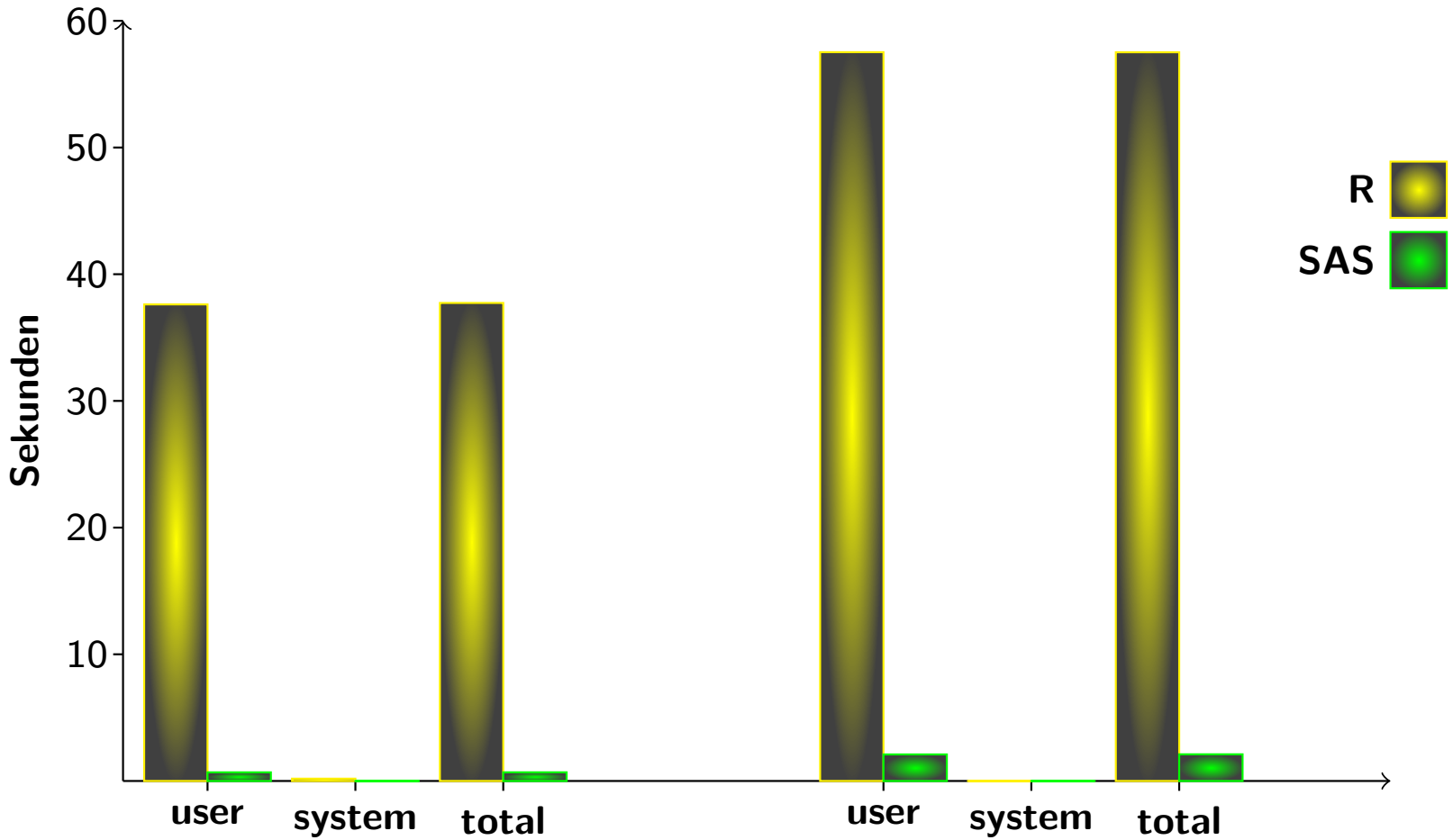
100 Mill. Wiederholungen



Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

Geschwindigkeitsvergleich Addition

100 Mill. Wiederholungen



Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

Intel Core 2 Quad CPU @2.4 GHz,
2GB RAM, 32 Bit-Betriebssystem

Erzeugen von Zufallszahlen

Erzeugen von Zufallszahlen

R

```
rnorm(100000000);
```

Erzeugen von Zufallszahlen

R

```
rnorm(100000000);
```

SAS

```
proc iml;
y=rand('NORMAL',0 ,
   repeat(1,100000000,1));
run;
```

Erzeugen von Zufallszahlen

R

```
rnorm(100000000);
```

SAS

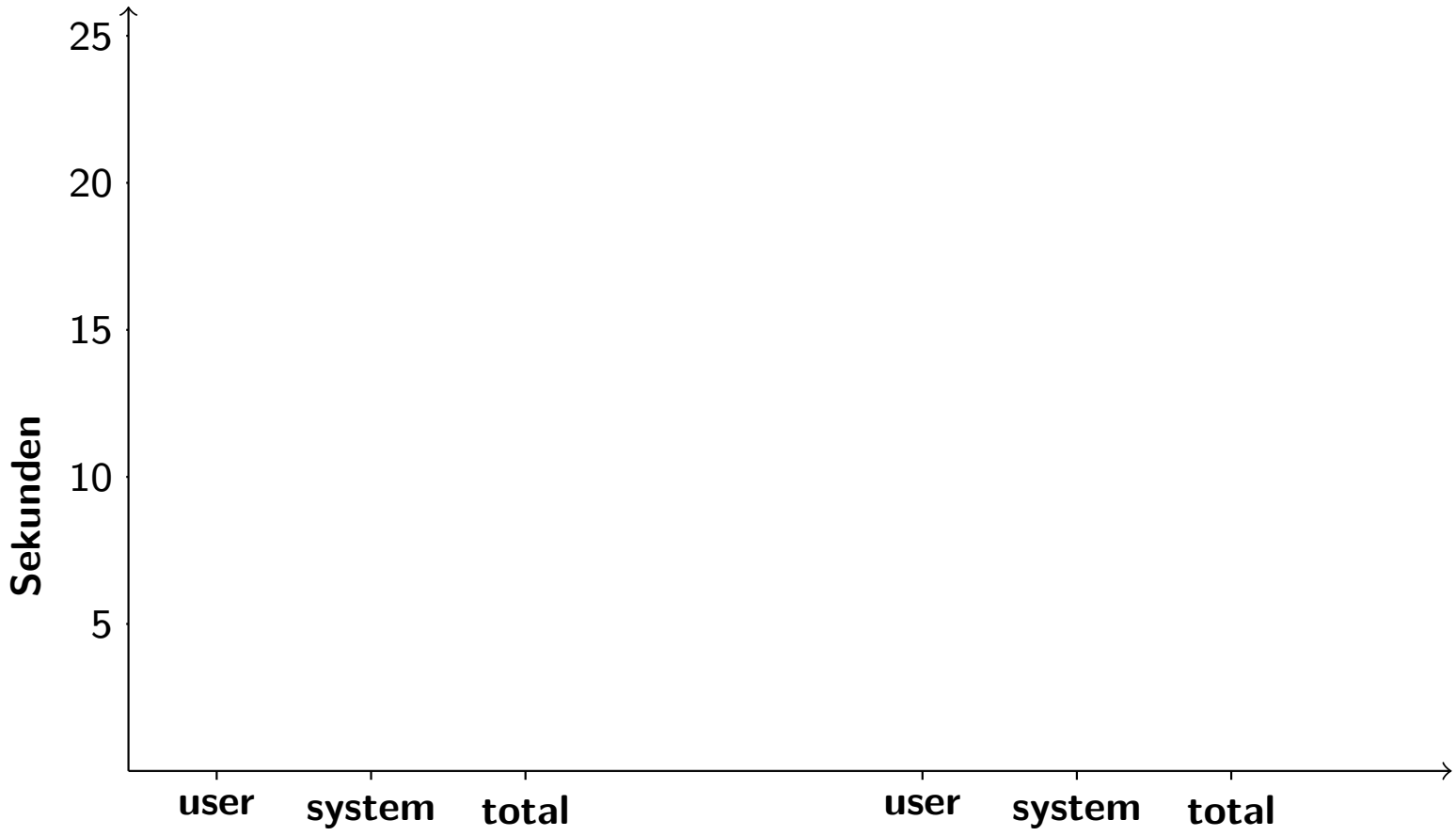
```
proc iml;
y=rand('NORMAL',0 ,
   repeat(1,100000000,1));
run;
```

Alternative:

```
data d0;
do i=1 to 100000000;
   x = rand('NORMAL', 0, 1);
   output;
end;
run;
```

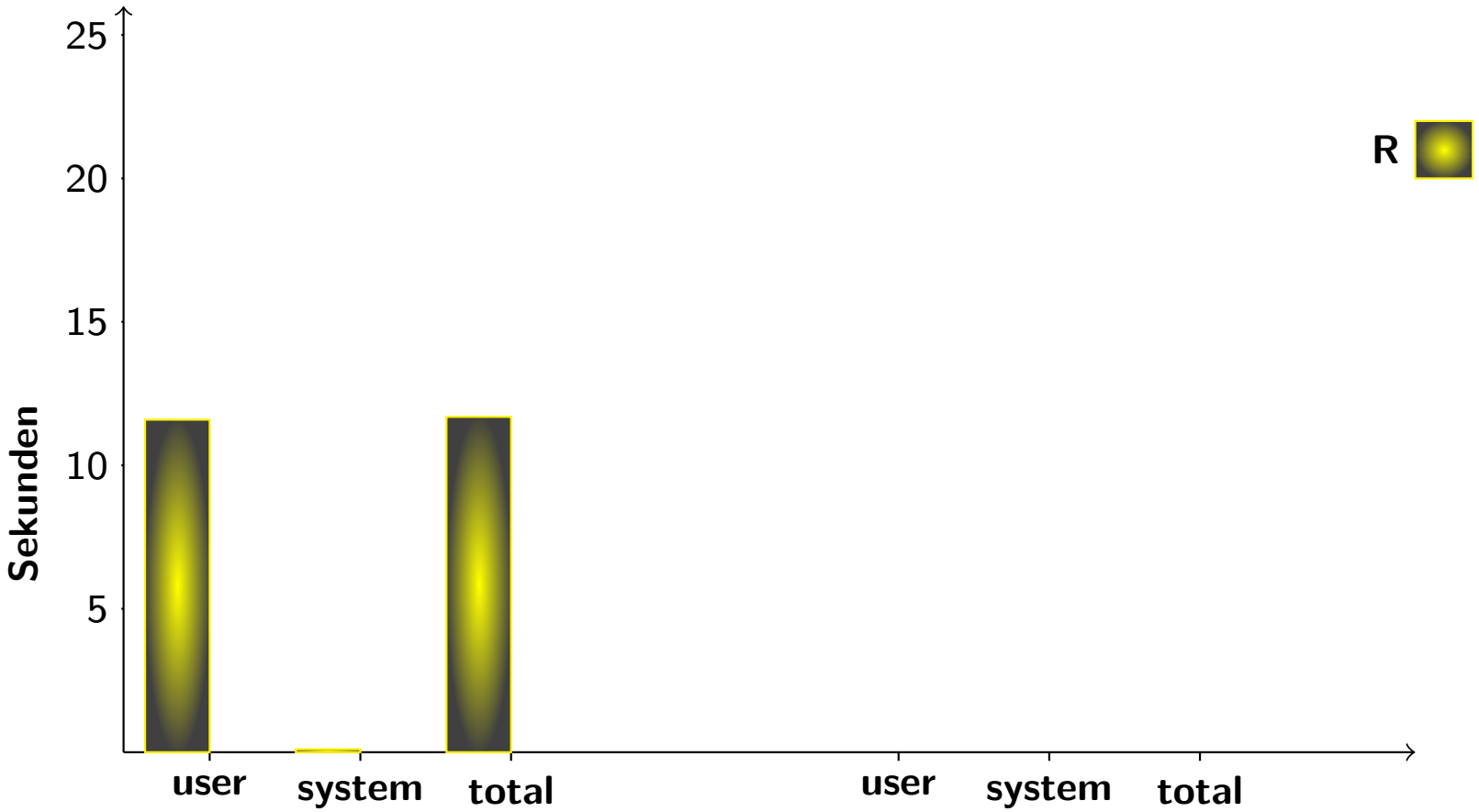
Erzeugen von Zufallszahlen

100 Mill. Zufallszahlen



Erzeugen von Zufallszahlen

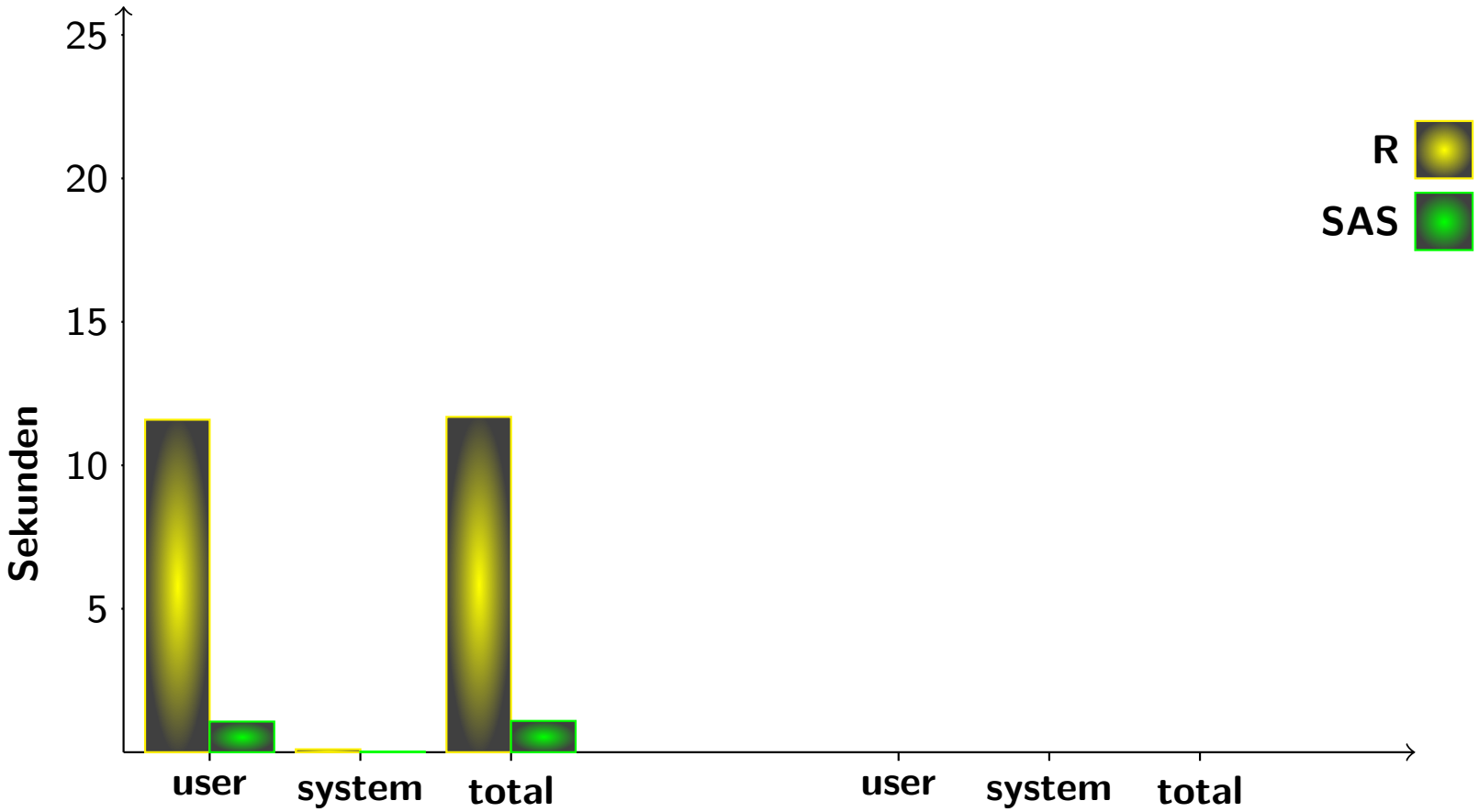
100 Mill. Zufallszahlen



Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

Erzeugen von Zufallszahlen

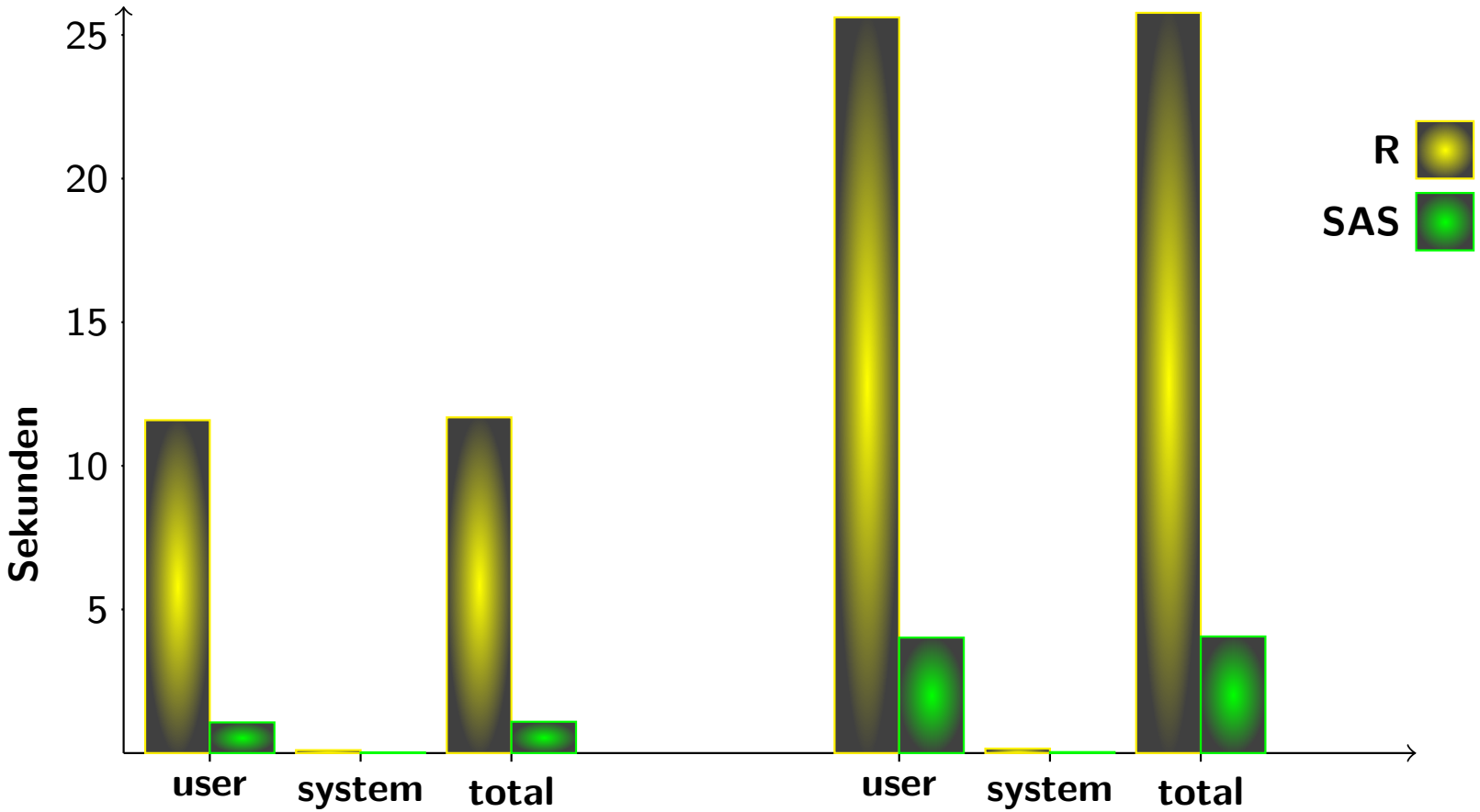
100 Mill. Zufallszahlen



Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

Erzeugen von Zufallszahlen

100 Mill. Zufallszahlen



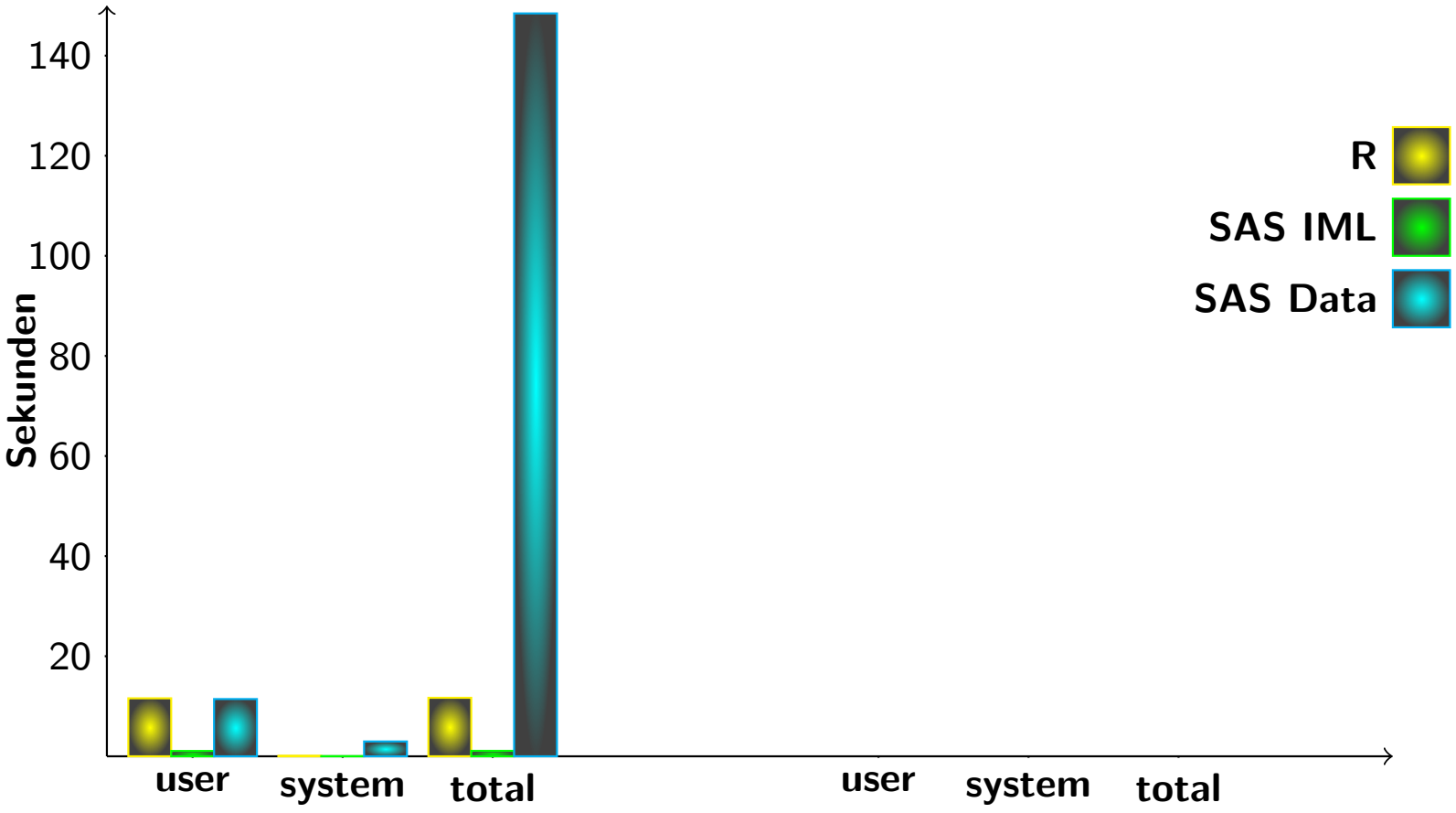
Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

Intel Core 2 Quad CPU @2.4 GHz,
2GB RAM, 32 Bit-Betriebssystem

Erzeugen von Zufallszahlen

Erzeugen von Zufallszahlen

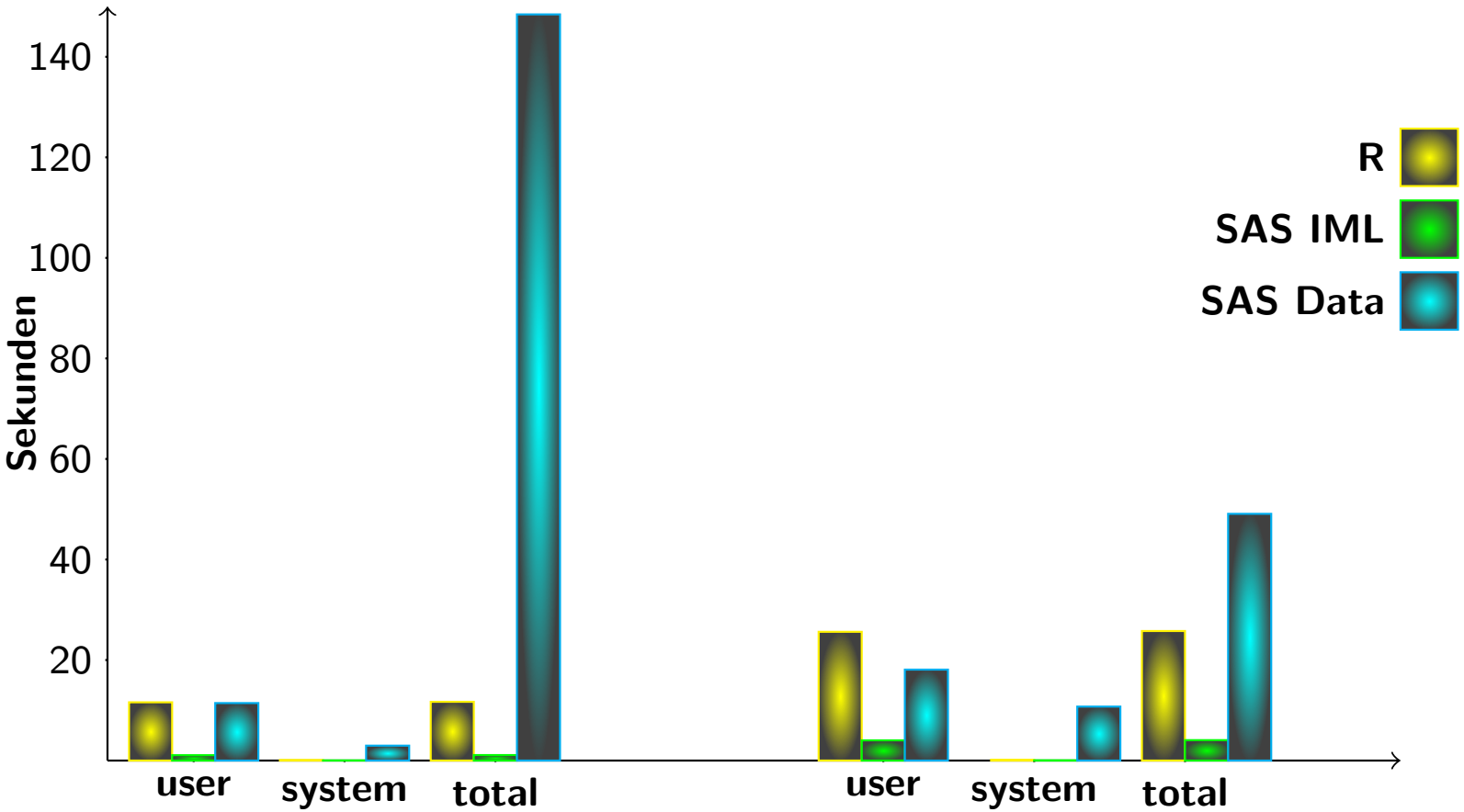
100 Mill. Zufallszahlen



Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

Erzeugen von Zufallszahlen

100 Mill. Zufallszahlen



Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

Intel Core 2 Quad CPU @2.4 GHz,
2GB RAM, 32 Bit-Betriebssystem

einfache Varianzanalyse

einfache Varianzanalyse

R

```
simAnova<-function(nsim,nf,nwh)
{
   :
fa <- factor(rep(1:nf,c(rep(nwh,nf))));
while (i < nsim)
{
   y<-c(rnorm(nwh,sd=1),
      rep(rnorm(nwh,sd=3),nf-1));
   df<-data.frame(y,fa);
   erg<-summary(lm(y~fa,data=df));
   p<-erg[[1]]$"Pr"[1];
   if (p<0.05){sum<-sum+1;}
   i<-i+1;
}
   :
}
system.time(simAnova(10000,2,5));
```

einfache Varianzanalyse

R

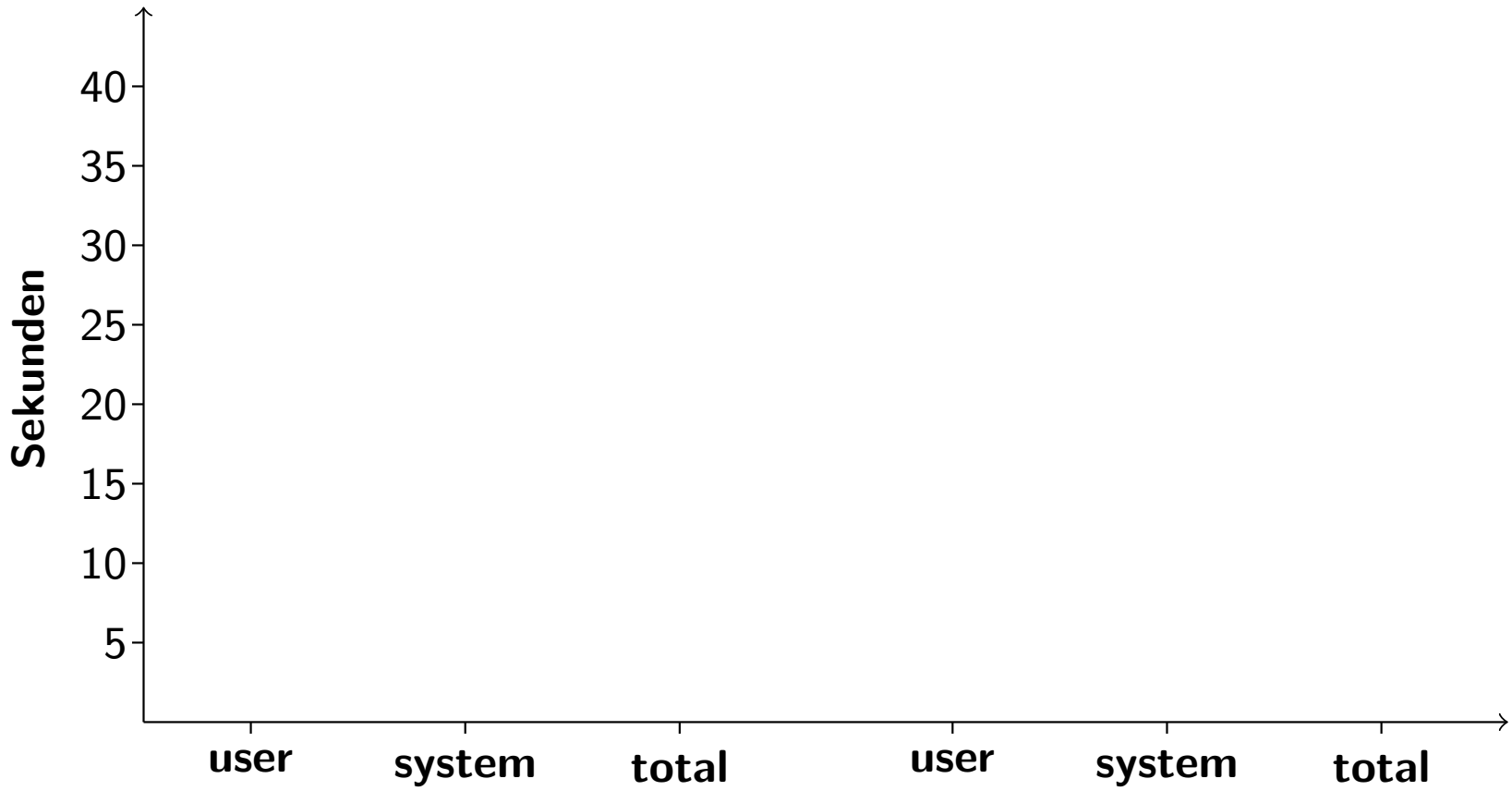
```
simAnova<-function(nsim,nf,nwh)
{
   :
fa <- factor(rep(1:nf,c(rep(nwh,nf))));
while (i < nsim)
{
   y<-c(rnorm(nwh,sd=1),
      rep(rnorm(nwh,sd=3),nf-1));
   df<-data.frame(y,fa);
   erg<-summary(lm(y~fa,data=df));
   p<-erg[[1]]$"Pr"[1];
   if (p<0.05){sum<-sum+1;}
   i<-i+1;
}
   :
}
system.time(simAnova(10000,2,5));
```

SAS

```
%macro sim (nsim,nf,nwh);
   :
data d0;
   do datensatz=1 to &nsim;
      do faktor=1 to &nf;
         do wh=1 to &nwh;
            if faktor=1 then x=rand('normal', 0, 1);
            else x=rand('normal', 0 , 3);
         output;
   end; end; end;
*sasfile d0 open;
proc anova data=d0 noprint outstat=d1;
   class faktor;
   model x=faktor;
   by datensatz;
   :
%mend sim;
%sim (10000,2,5); run;
```

einfache Varianzanalyse

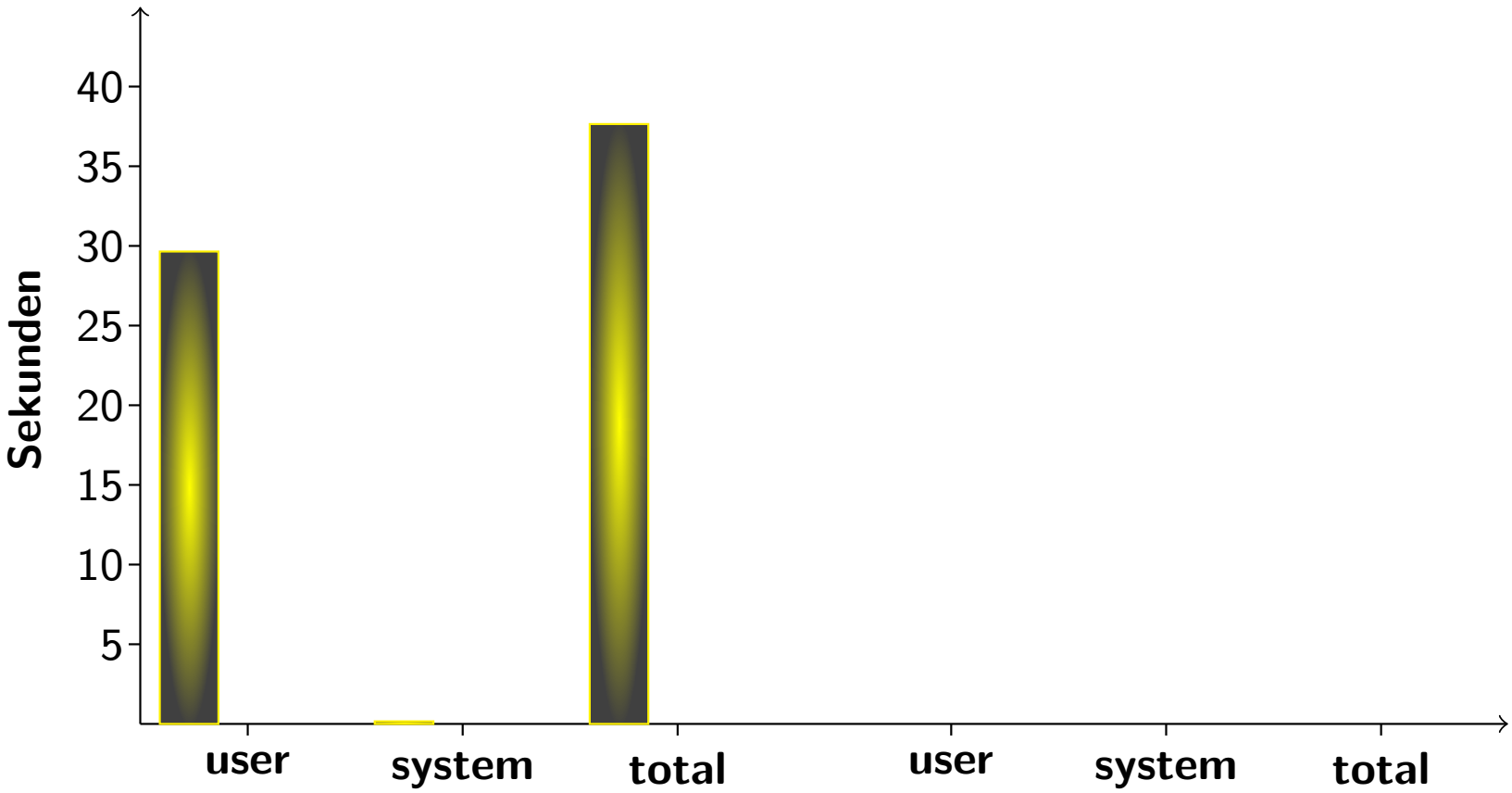
10 000 Simulationsläufe



einfache Varianzanalyse

10 000 Simulationsläufe

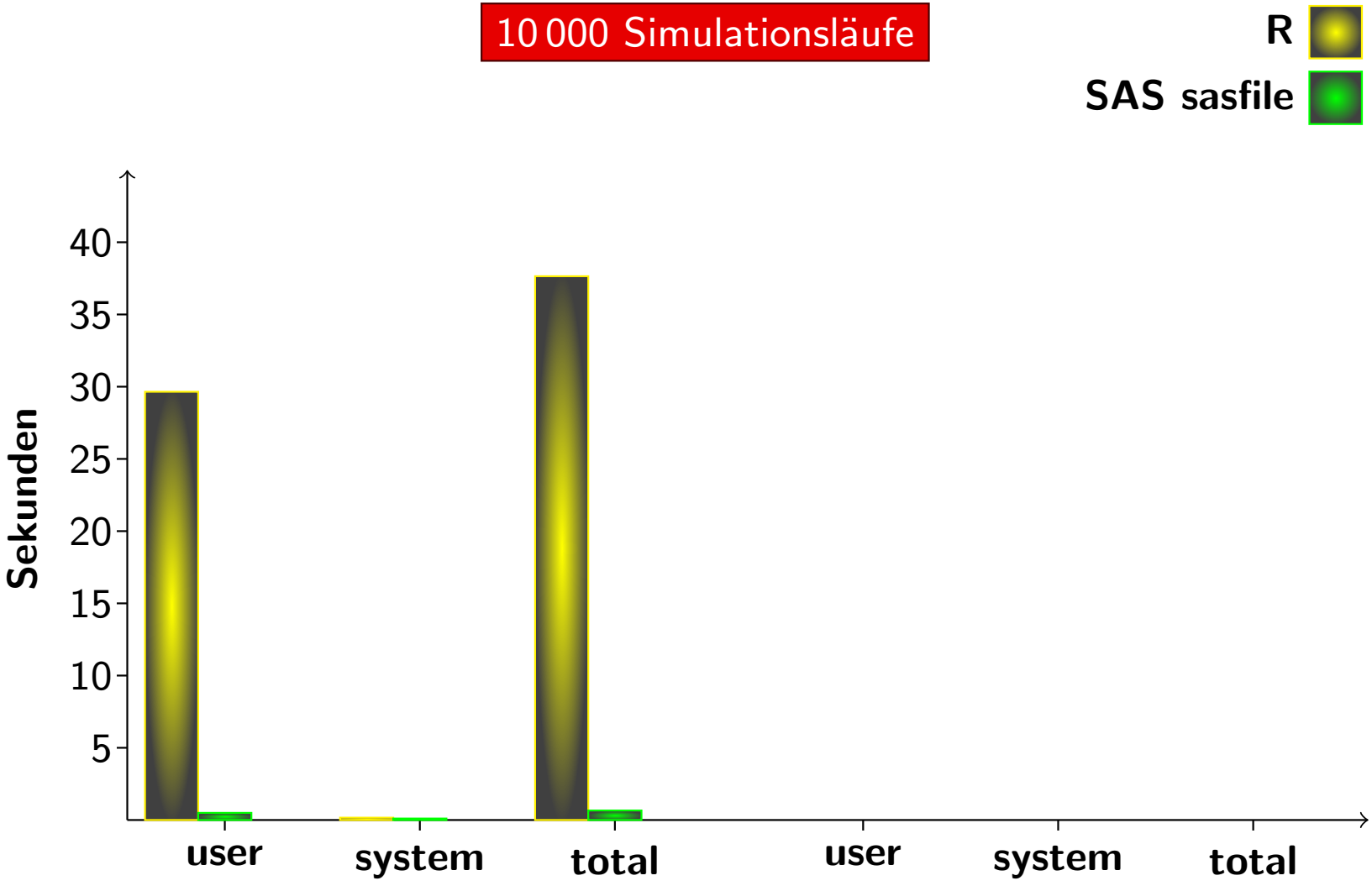
R



Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

einfache Varianzanalyse

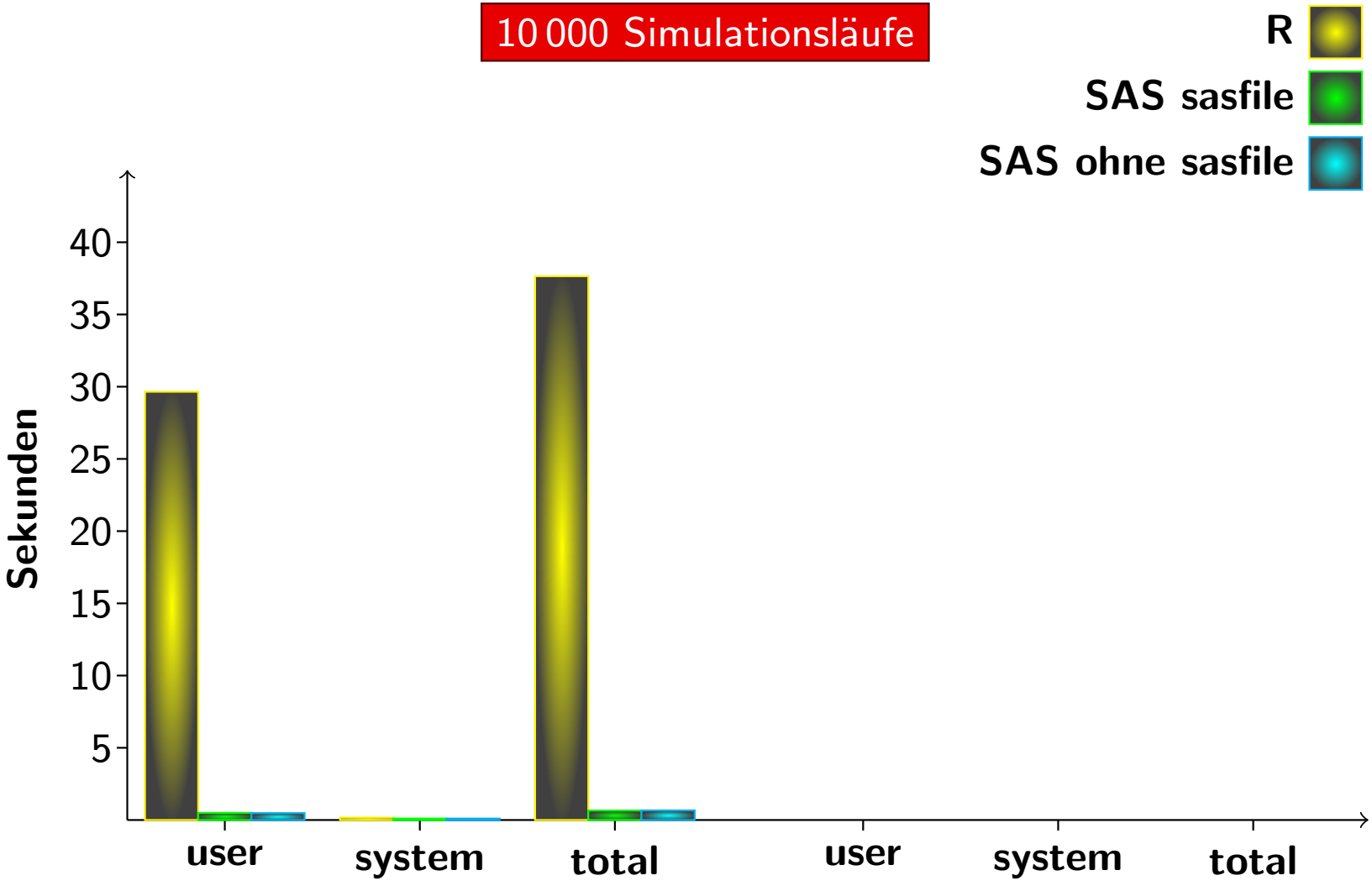
10 000 Simulationsläufe



Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

einfache Varianzanalyse

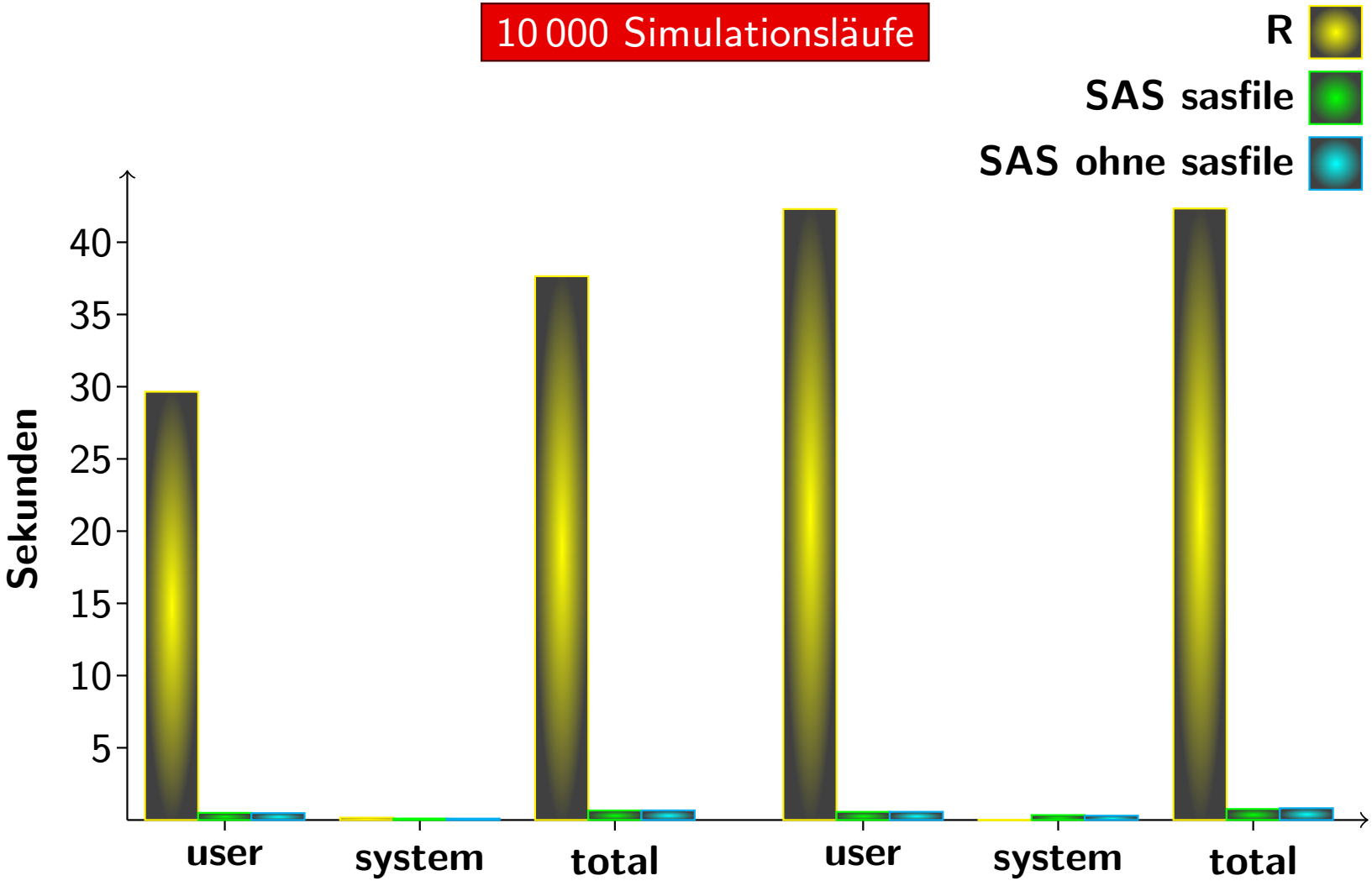
10 000 Simulationsläufe



Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

einfache Varianzanalyse

10 000 Simulationsläufe



Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

Intel Core 2 Quad CPU @2.4 GHz,
2GB RAM, 32 Bit-Betriebssystem

einfache Varianzanalyse

Ergebnisse

einfache Varianzanalyse

Ergebnisse

Vorgaben:

Faktorstufen: 2

Wiederholungen: 5

$$\mu_1 = \mu_2 = 0$$

$$\sigma_1 = 1,\ \sigma_2 = 3$$

$$\alpha = 0.05$$

einfache Varianzanalyse

Ergebnisse

Vorgaben:

Faktorstufen: 2

Wiederholungen: 5

$\mu_1 = \mu_2 = 0$

$\sigma_1 = 1, \sigma_2 = 3$

$\alpha = 0.05$

Simulationsergebnis:

realisiertes α=0.067

Einfache VA mit Matrixalgebra

Einfache VA mit Matrixalgebra

R

```
aov_malg<-function(nsim,nf,nwh)
{
library(MASS);
x<- array(c(rep(1,3*nwh),rep(0,2*nwh),
   rep(1,nwh)),dim=c(2*nwh,3));
dff=ncol(x)-2; dfe=nrow(x)-dff-1;
xsxi=ginv(t(x)%*%x);
hat=x%*%xsxi%*%t(x);
h=diag(2*nwh)-hat;
n=0; i=0;
while (i < nsim)
{
   y<-c(rnorm(nwh,sd=1),rep(rnorm(nwh,sd=3),nf-1))
   xsy=t(x)%*%y;
   b=xsxi%*%xsy;
   r=h%*%y;
   sse=sum(r*r);
   mse=sse/dfe;
   k=sum(y);
   m=k*k/nrow(x);
   ssf=t(b)%*%t(x)%*%y-m;
   msf=ssf/dff;
   f=msf/mse;
   prob=1-pf(f,dff,dfe);
   if (prob <0.05) {n=n+1};
   i=i+1;
}
erg=n/nsim;
print (erg);
}
aov_malg(1000000,2,5);
```

Einfache VA mit Matrixalgebra

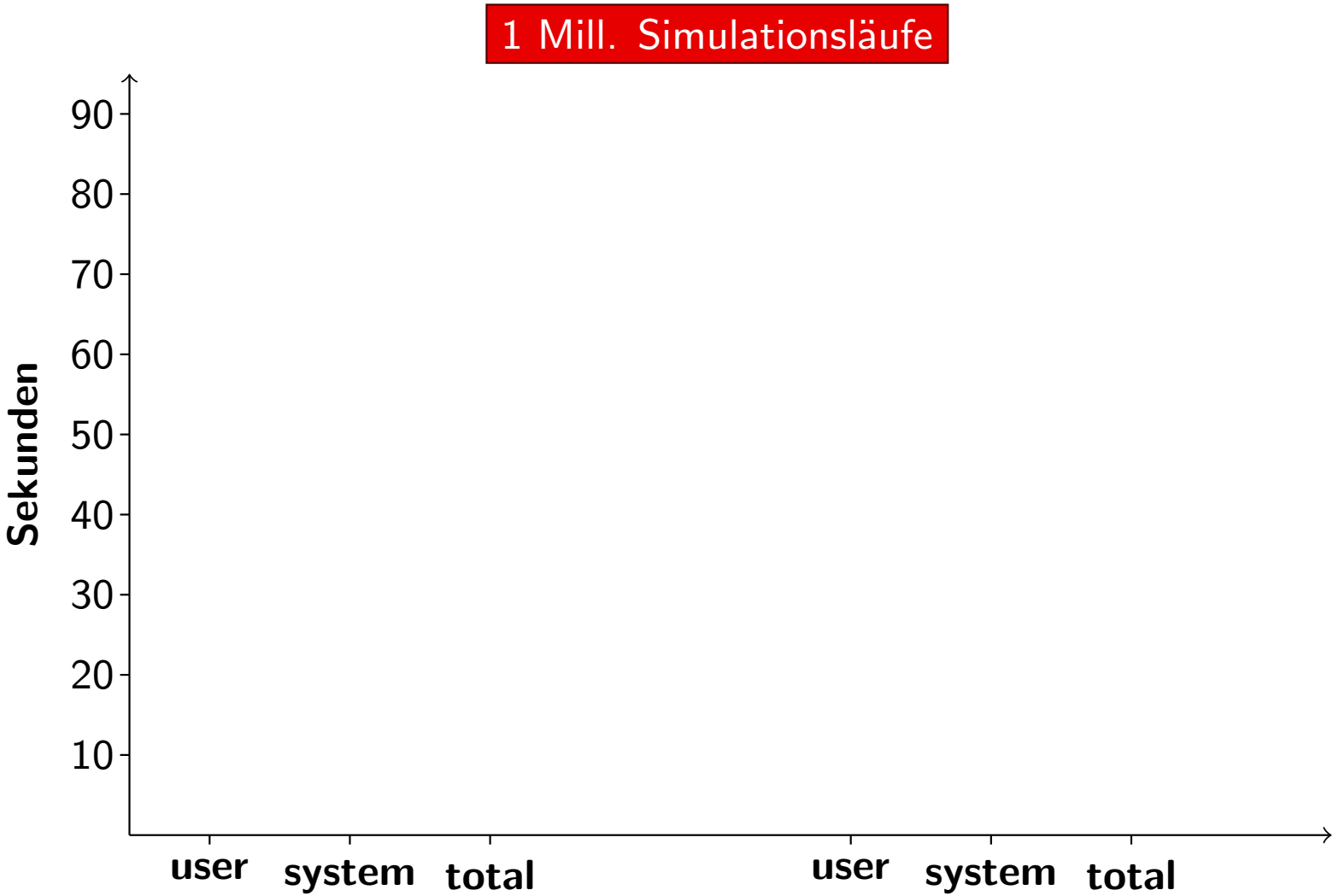
R

```
aov_malg<-function(nsim,nf,nwh)
{
library(MASS);
x<- array(c(rep(1,3*nwh),rep(0,2*nwh),
   rep(1,nwh)),dim=c(2*nwh,3));
dff=ncol(x)-2; dfe=nrow(x)-dff-1;
xsxi=ginv(t(x)%*%x);
hat=x%*%xsxi%*%t(x);
h=diag(2*nwh)-hat;
n=0; i=0;
while (i < nsim)
{
   y<-c(rnorm(nwh,sd=1),rep(rnorm(nwh,sd=3),nf-1))
   xsy=t(x)%*%y;
   b=xsxi%*%xsy;
   r=h%*%y;
   sse=sum(r*r);
   mse=sse/dfe;
   k=sum(y);
   m=k*k/nrow(x);
   ssf=t(b)%*%t(x)%*%y-m;
   msf=ssf/dff;
   f=msf/mse;
   prob=1-pf(f,dff,dfe);
   if (prob <0.05) {n=n+1};
   i=i+1;
}
erg=n/nsim;
print (erg);
}
aov_malg(1000000,2,5);
```

SAS

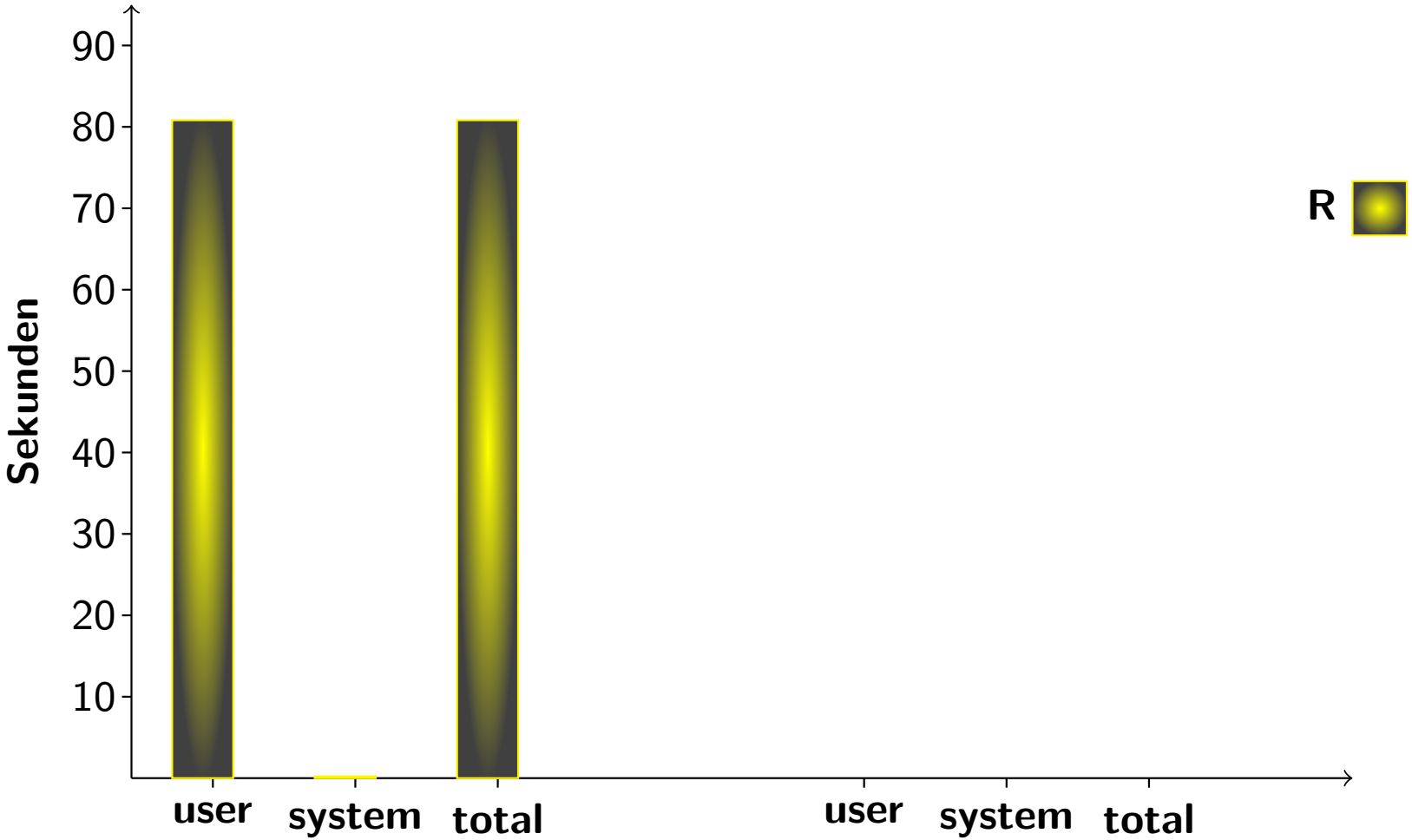
```
%macro aov_malg (nsim,nf,nwh);
proc iml;
nsim=%eval(&nsim); nwh=%eval(&nwh);
x=&x;
dff=ncol(x)-2;
dfe=nrow(x)-dff-1;
xsxi=ginv(x'*x);
hat=x*xsxi*x';
h=i(2*nwh)-hat;
n=0;
do i=1 to nsim;
   y=rand('normal', 0 ,repeat(1,nwh,1))
     //rand('normal', 0 ,repeat(3,nwh,1));
   xsy=x'*y;
   b=xsxi*xsy;
   r=h*y;
   sse=ssq(r);
   mse=sse/dfe;
   k=sum(y);
   m=k*k/nrow(x);
   ssf=b'*x'*y-m;
   msf=ssf/dff;
   f=msf/mse;
   prob=1-probf(f,dff,dfe);
   if prob <0.05 then n=n+1;
end;
erg=n/nsim;
print erg;
%end; %mend aov_malg;
%aov_malg(1000000,2,5);run;
```

Einfache VA mit Matrixalgebra



Einfache VA mit Matrixalgebra

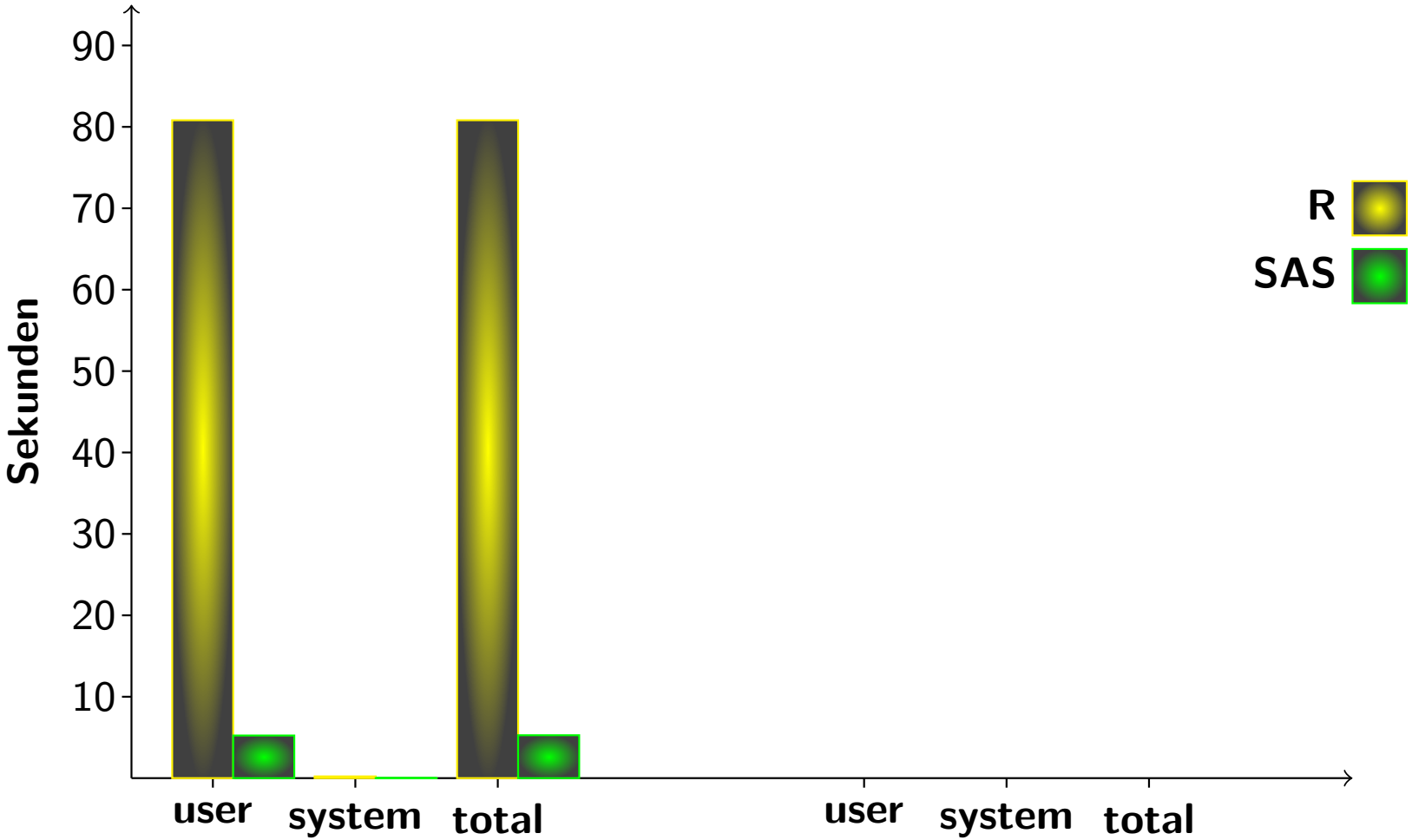
1 Mill. Simulationsläufe



Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

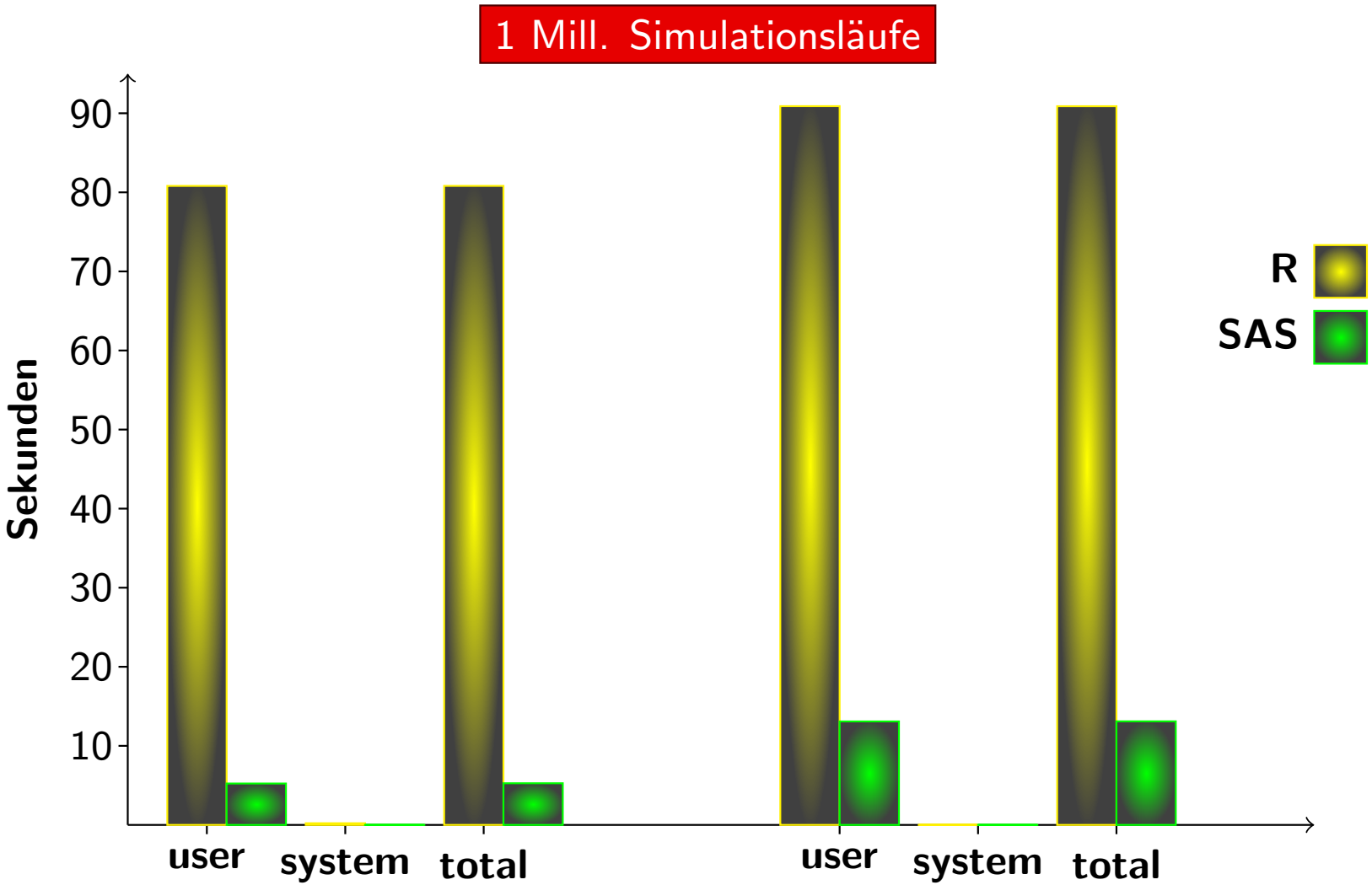
Einfache VA mit Matrixalgebra

1 Mill. Simulationsläufe



Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

Einfache VA mit Matrixalgebra



Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

Intel Core 2 Quad CPU @2.4 GHz,
2GB RAM, 32 Bit-Betriebssystem

zweifaktorielle Varianzanalyse

zweifaktorielle Varianzanalyse

R

```
simAov2f<-function(nsim,nf1,nf2,nwh,typ)
{
      .
      .
      .
   fa2 <- factor(rep(rep(1:nf2,c(rep(nwh,nf2))),nf1));
   fa1 <- factor(rep(1:nf1,c(rep(nwh*nf2,nf1))));
   for (i in 1:nsim)
   {
   y=c();
      for (f1 in 1:nf1)
      {
         for (f2 in 1:nf2)
         {
            if (typ==1) {yh <- c(rnorm(nwh,sd=f1));}
            else if (typ==2) {yh <- c(rnorm(nwh,sd=f2));}
            else {yh <- c(rnorm(nwh,sd=f1+f2));}
            y <- c(y,yh);
         }
      }
   df<-data.frame(y,fa1,fa2);
   erg<- anova(lm(y fa1+fa2+fa1:fa2,data=df));
   }

   .
   .
   .
}
system.time(simAov2f(10000,2,3,5,1));
```

zweifaktorielle Varianzanalyse

R

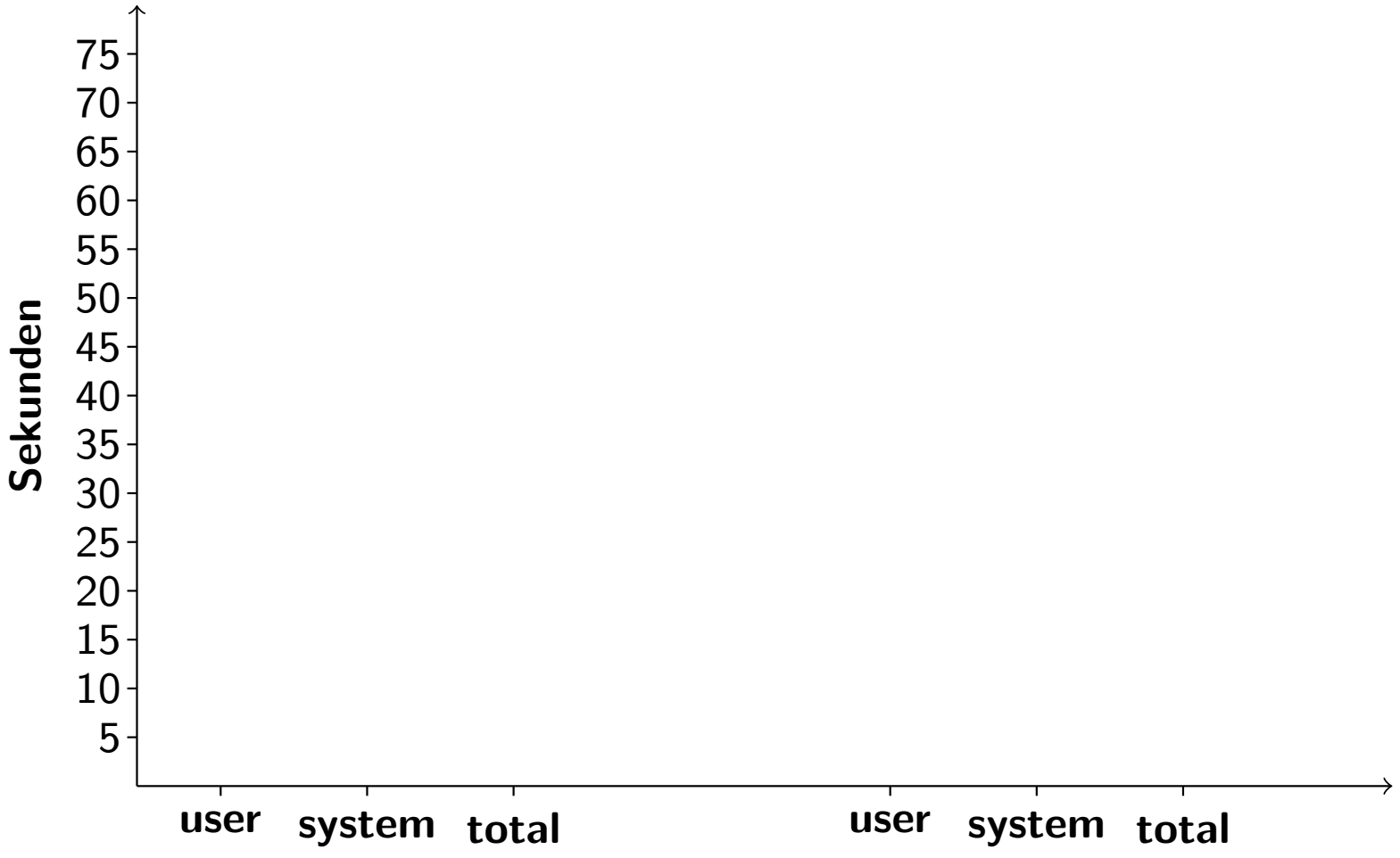
```
simAov2f<-function(nsim,nf1,nf2,nwh,typ)
{
      .
      .
      .
   fa2 <- factor(rep(rep(1:nf2,c(rep(nwh,nf2))),nf1));
   fa1 <- factor(rep(1:nf1,c(rep(nwh*nf2,nf1))));
   for (i in 1:nsim)
   {
   y=c();
      for (f1 in 1:nf1)
      {
         for (f2 in 1:nf2)
         {
            if (typ==1) {yh <- c(rnorm(nwh,sd=f1));}
            else if (typ==2) {yh <- c(rnorm(nwh,sd=f2));}
            else {yh <- c(rnorm(nwh,sd=f1+f2));}
            y <- c(y,yh);
         }
      }
   df<-data.frame(y,fa1,fa2);
   erg<- anova(lm(y fa1+fa2+fa1:fa2,data=df));
   }
   .
   .
   .
}
system.time(simAov2f(10000,2,3,5,1));
```

SAS

```
%macro simAov2f (nsim,nf1,nf2,nwh,typ);
   .
   .
   .
data d0;
   do datensatz=1 to &nsim;
      do f1=1 to &nf1;
         do f2=1 to &nf2;
            do wh=1 to &nwh;
               select(%eval(&typ));
               when (1) x = rand('normal', 0, f1);
               when (2) x = rand('normal', 0, f2);
               otherwise x = rand('normal', 0, f1+f2);
            end;
            output;
   end; end; end; end;
*sasfile d0 open;
proc anova data=d0 noprint outstat=d1;
   class f1 f2;
   model x=f1 f2 f1*f2;
   by datensatz;
*sasfile d0 close;
   .
   .
   .
%mend sim;
%simAov2f (10000,2,3,5,1); run;
```

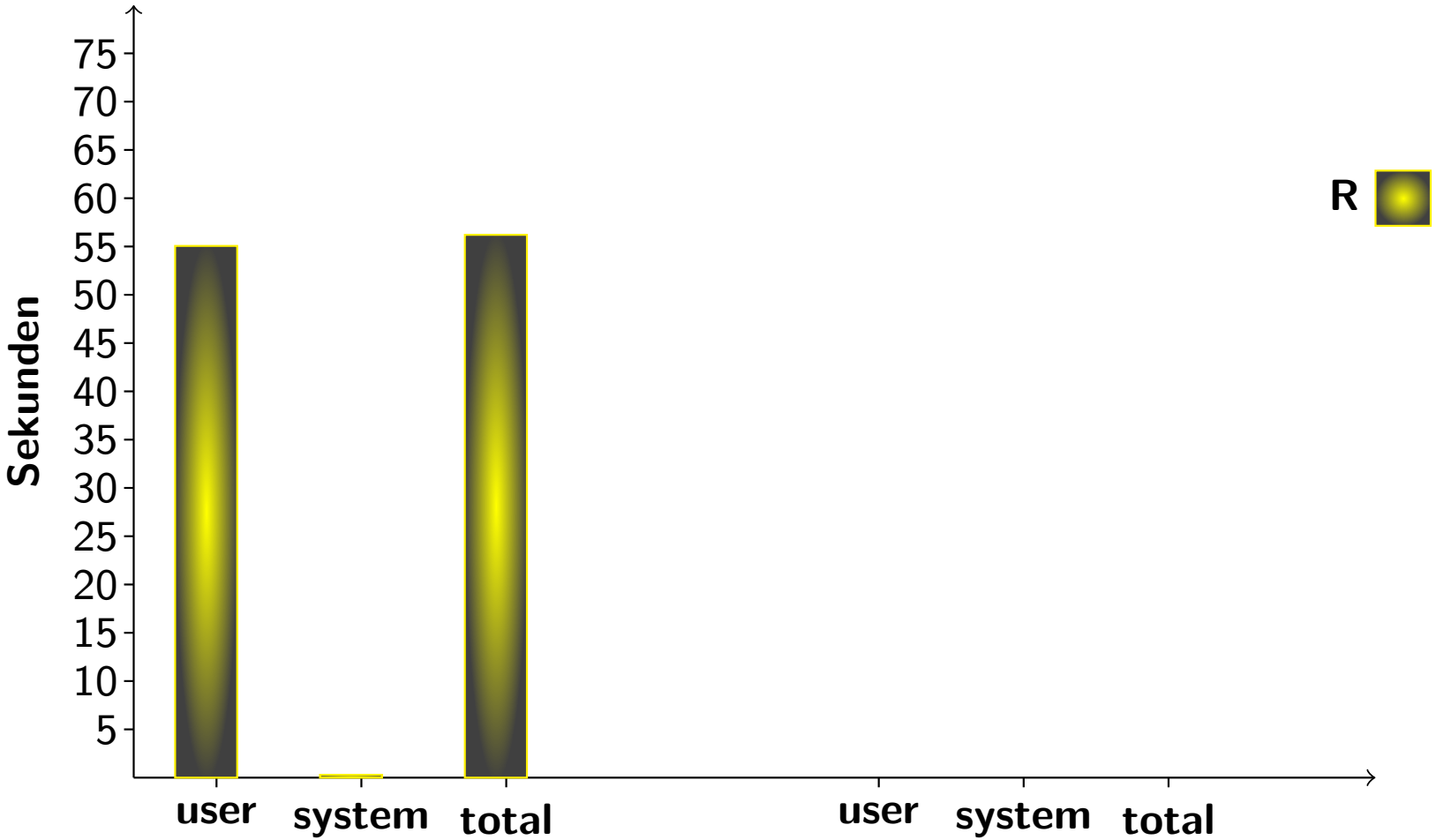
zweifaktorielle Varianzanalyse

10 000 Simulationsläufe



zweifaktorielle Varianzanalyse

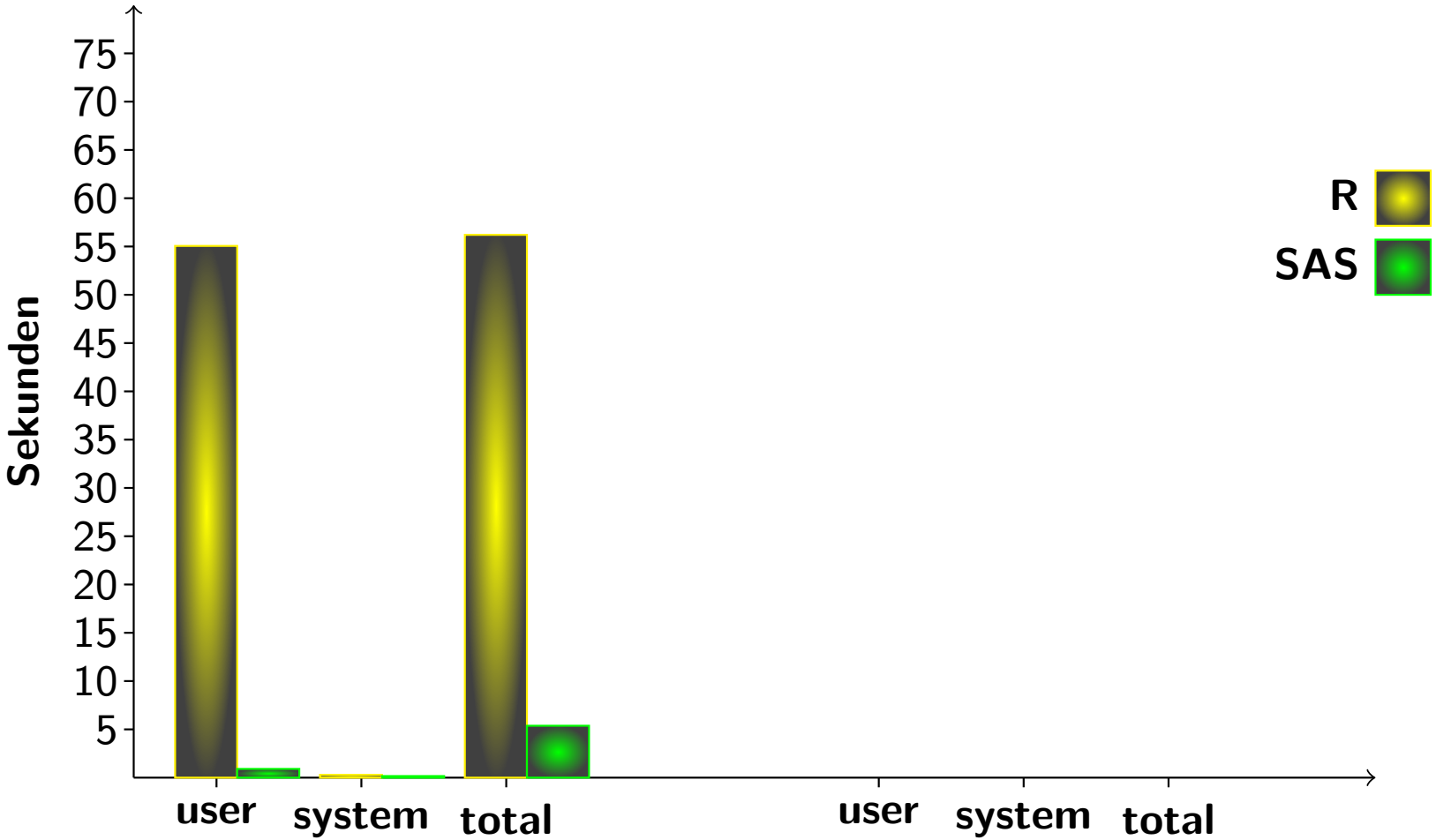
10 000 Simulationsläufe



Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

zweifaktorielle Varianzanalyse

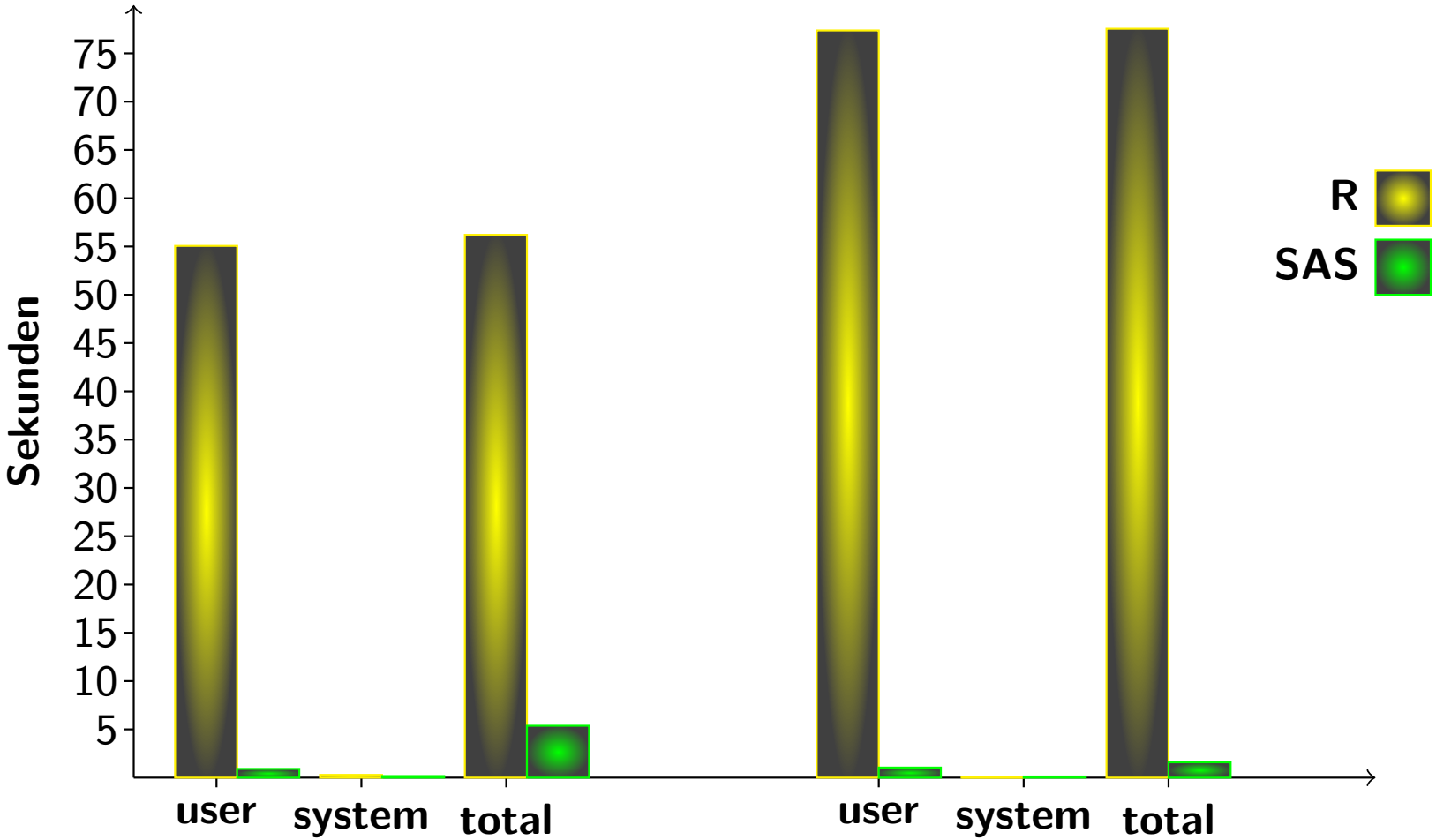
10 000 Simulationsläufe



Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

zweifaktorielle Varianzanalyse

10 000 Simulationsläufe



Intel Core i7/ CPU 965 @3.2 GHz,
8.18 GB RAM, 64 Bit Betriebssystem

Intel Core 2 Quad CPU @2.4 GHz,
2GB RAM, 32 Bit-Betriebssystem

zweifaktorielle Varianzanalyse

Vorgaben:

Stufen Faktor1: 2 Stufen Faktor1: 3 Wiederholungen: 5
$\mu_{11} = \mu_{12} = \mu_{13} = \mu_{21} = \mu_{22} = \mu_{23} = 0$ α=0.05

$\sigma_{11} = \sigma_{12} = \sigma_{13}$	1	$\sigma_{11} = \sigma_{21} =$	1
$\sigma_{21} = \sigma_{23} = \sigma_{23}$	2	$\sigma_{12} = \sigma_{22} =$	2
		$\sigma_{13} = \sigma_{23} =$	3

zweifaktorielle Varianzanalyse

Vorgaben:

Stufen Faktor1: 2
Stufen Faktor1: 3
Wiederholungen: 5

$$\mu_{11} = \mu_{12} = \mu_{13} = \mu_{21} = \mu_{22} = \mu_{23} = 0$$

α=0.05

$\sigma_{11} = \sigma_{12} = \sigma_{13}$	1	$\sigma_{11} = \sigma_{21} =$	1
$\sigma_{21} = \sigma_{23} = \sigma_{23}$	2	$\sigma_{12} = \sigma_{22} =$	2
		$\sigma_{13} = \sigma_{23} =$	3

realisiertes α

Faktor 1:	0.0539	0.0551
Faktor 2:	0.0546	0.0644
Wechselwirkung:	0.0560	0.0638

Mittelwerte der Geschwindigkeitsmessung

Intel Core 2 Quad CPU @2.4 GHz, 2GB RAM
32 Bit-Betriebssystem Windows XP

	R			SAS			
	user	system	total	user	system	total	n
Addition	98.740	0.123	98.874	26.788	0.004	26.802	100 Mill.
IF	104.707	0.155	104.884	25.483	0.005	25.634	100 Mill.
Wertzuweisung	41.213	0.135	41.354	24.691	0.009	24.695	100 Mill.
Zufallszahlen	25.612	0.156	25.771	4.021	0.034	4.060	100 Mill.
schleife	18.740	0.136	18.881	3.201	0.001	3.202	100 Mill.
Zfz write				18.091	10.735	49.082	100 Mill.
einf.VA apply	43.175	0.031	45.854				10 000
einf.VA	42.301	0.008	42.336	5.552	0.377	7.196	10 000
Matrixalgebra	90.899	0.014	90.892	13.089	0.002	13.101	1 Mill.
2-fakt.VA	77.369	0.021	77.547	1.043	0.119	1.585	10 000

Mittelwerte der Geschwindigkeitsmessung

Intel(R) Core i/CPU 965 @3.2 GHz, 8.18 GB RAM
64 Bit Betriebssystem, Windows Vista Business

	R			SAS			
	user	system	total	user	system	total	n
Schleifen	14.8360	0.0820	14.9480	1.0370	0.0000	1.0370	100 Mill.
IF	78.9260	0.2370	79.6180	11.6010	0.0010	11.6080	100 Mill.
Zufallszahlen	11.5890	0.1020	11.6880	1.0680	0.0170	1.0910	100 Mill.
Wertzuweisung (k=1)	33.2610	0.1160	33.4820	9.6210	0.0040	9.6280	100 Mill.
Addition (k=i+i)	70.8780	0.1830	71.2140	10.3040	0.0010	10.3030	100 Mill.
VA Matrixalgebra	80.8130	0.2120	80.8160	5.2370	0.0000	5.2560	1 Mill.
VA2f (1.Fakt inhom.)	55.6020	0.2860	56.2070	0.9050	0.1631	4.7440	10 000
VA2f (2.Fakt inhom.)	55.6520	0.2670	56.3520	0.8800	0.1680	2.4850	10 000
VA for (aov)	38.4160	0.1670	38.6680				10 000
VA while (aov)	36.9110	0.1600	37.6480				10 000
VA for (lm)	33.6830	0.1460	34.0370				10 000
VA repeat (lm)	29.2300	0.1620	29.6490				10 000
VA apply (lm)	30.0950	0.1300	30.4420				10 000
Zfz ohne speichern				6.8790	0.0020	6.8730	100 Mill.
Zfz mit speichern				11.4380	2.9650	148.4400	100 Mill.
Zfz mit IML				10.8970	0.2770	11.1860	100 Mill.
VA mit "sasfile"				5.0100	1.0100	26.9080	100 000
VA mit "sasfile"				0.5230	0.0970	0.6130	10 000
VA ohne "sasfile"				5.0580	1.0510	77.0960	100 000
VA ohne "sasfile"				0.5040	0.0990	0.6100	10 000

Mittelwerte der Geschwindigkeitsmessung

Intel(R) Pentium M Processor 1.73 Ghz, 504 MB RAM
Microsoft Windows XP Professional, Version 2002, Service Pack 3

	R			SAS			
	user	system	total	user	system	total	n
Addition	74.4650	0.1340	83.6260	0.9720	0.0001	0.8490	100 Mill.
IF	124.0400	0.0100	124.5960	38.1870	0.0001	38.2430	100 Mill.
Schleife	32.4120	0.4530	33.9520	5.7000	0.0030	5.7460	100 Mill.
Wertzuweisung	29.0640	0.0001	29.1370	33.2350	0.0030	33.4190	100 Mill.
Zufallszahlen	3.1110	0.0580	3.1760	5.0560	0.1400	5.2870	10 Mill.
VA (Matrixalgebra)	133.6040	0.1420	135.7600	18.9470	0.0030	19.0690	1 Mill.
einfache VA	70.4010	0.0640	71.3180				10 000
einfache VA(apply)	62.9720	0.0800	63.9620				10 000
einfache VA				0.8864	0.0491	1.0873	10 000
2 fakt. VA	112.9080	0.1150	114.9390	1.8127	0.2027	3.4073	10 000

Standardabweichung der Geschwindigkeitsmessung

Intel(R) Pentium M Processor 1.73 Ghz, 504 MB RAM
Microsoft Windows XP Professional, Version 2002, Service Pack 3

	R			SAS			
	user	system	total	user	system	total	n
Addition	0.70845	0.23029	0.37924	0.11499	0.00527	0.25704	100 Mill.
IF	0.13028	0.08957	1.19826	0.09787	0.00316	0.25723	100 Mill.
Schleife	1.22586	0.20282	1.83020	0.03464	0.00483	0.08720	100 Mill.
Wertzuweisung	0.16447	0.05720	1.55171	0.10047	0.00516	0.39562	100 Mill.
Zufallszahlen	0.03479	0.02394	0.05481	0.06653	0.04667	0.30660	10 Mill.
VA (Matrixalgebra)	0.44023	0.06070	0.53864	0.10414	0.00483	0.24131	1 Mill.
einfache VA	0.12914	0.03777	0.64720				10 000
einfache VA apply	0.33296	0.04989	0.80235				10 000
einfache VA				0.03139	0.02468	0.13116	10 000
2 fakt. VA	0.40038	0.04743	2.06284	0.12240	0.05101	1.25773	10 000

Abschließende Bemerkungen

- SAS ist bei Simulationen schneller als R.
- Bei langsamen Maschinen Unterschied zwischen R und SAS etwas kleiner.
- Die Simulation mit selbst entwickelten Programmen (Matrixalgebra) ist deutlich schneller, als die Simulation mit der wiederholten Anwendung vorgefertigter Routinen.
- R und SAS wachsen zusammen (R kann aus SAS heraus aufgerufen werden).
- (Programme in Fortran, C, . . . sind wesentlich schneller als solche in SAS, R . . .).

Abschließende Bemerkungen

- SAS ist bei Simulationen schneller als R.
- Bei langsamen Maschinen Unterschied zwischen R und SAS etwas kleiner.
- Die Simulation mit selbst entwickelten Programmen (Matrixalgebra) ist deutlich schneller, als die Simulation mit der wiederholten Anwendung vorgefertigter Routinen.
- R und SAS wachsen zusammen (R kann aus SAS heraus aufgerufen werden).
- (Programme in Fortran, C, . . . sind wesentlich schneller als solche in SAS, R . . .).

Abschließende Bemerkungen

- SAS ist bei Simulationen schneller als R.
- Bei langsamen Maschinen Unterschied zwischen R und SAS etwas kleiner.
- Die Simulation mit selbst entwickelten Programmen (Matrixalgebra) ist deutlich schneller, als die Simulation mit der wiederholten Anwendung vorgefertigter Routinen.
- R und SAS wachsen zusammen (R kann aus SAS heraus aufgerufen werden).
- (Programme in Fortran, C, . . . sind wesentlich schneller als solche in SAS, R . . .).

Abschließende Bemerkungen

- SAS ist bei Simulationen schneller als R.
- Bei langsamen Maschinen Unterschied zwischen R und SAS etwas kleiner.
- Die Simulation mit selbst entwickelten Programmen (Matrixalgebra) ist deutlich schneller, als die Simulation mit der wiederholten Anwendung vorgefertigter Routinen.
- R und SAS wachsen zusammen (R kann aus SAS heraus aufgerufen werden).
- (Programme in Fortran, C, . . . sind wesentlich schneller als solche in SAS, R . . .).

Abschließende Bemerkungen

- SAS ist bei Simulationen schneller als R.
- Bei langsamen Maschinen Unterschied zwischen R und SAS etwas kleiner.
- Die Simulation mit selbst entwickelten Programmen (Matrixalgebra) ist deutlich schneller, als die Simulation mit der wiederholten Anwendung vorgefertigter Routinen.
- R und SAS wachsen zusammen (R kann aus SAS heraus aufgerufen werden).
- (Programme in Fortran, C, . . . sind wesentlich schneller als solche in SAS, R . . .).